NISTIR 7302

Proceedings of the 2005 Conference on IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

Kang B. Lee
National Institute of Standards and Technology

John C. Eidson
Agilent Technologies

Hans Weibel
Zurich University of Applied Sciences

Dirk Mohl
Hirschmann Automation and Control

December 2005

Proceedings of the 2005 Conference on IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

Co-sponsored by
The National Institute of Standards and Technology
The Instrumentation and Measurement Society
The Institute of Electrical and Electronics Engineers

Zurich University of Applied Sciences
Winterthur, Switzerland
October 10 – 12, 2005

Conference Co-Chairs: John Eidson, Agilent Technologies
Kang Lee, NIST
Hans Weibel, Zurich University of Applied Sciences
Plug-fest Chair: Dirk Mohl, Hirschmann Automation and Control

Disclaimer
This publication consists of workshop proceedings containing presentation slides, technical papers, recommendations, and other materials contributed by participants of this workshop. This publication provides the material as presented and discussed at the workshop in its original form, without modification by the National Institute of Standards and Technology (NIST). Commercial equipment and software referred to in this document are identified for informational purposes only, and does not imply recommendations of or endorsement by NIST, nor does it imply that the products so identified are the best available for the purpose.

TABLE OF CONTENTS

AGENDA 4
EXECUTIVE SUMMARY 7
PRESENTATIONS 8

TECHNICAL PRESENTATION—IEEE 1588 TUTORIAL

IEEE 1588 Basics -
John C. Eidson, Agilent Technologies 9
Telecommunications Applications –
Silvana Rodrigues, Zarlink Semiconductor; Antti Pietilainen, Nokia 62
Industrial and Motion Control Applications –
Anatoly Moldovansky, Rockwell Automation; Ludwig Winkel, Siemens 73
Test and Measurement Applications –
John C. Eidson, Agilent Technologies 88

TECHNICAL PRESENTATION-SESSION I – IMPLEMENTATION TOPICS-1

Design Considerations for Software-only Implementations of the Precision Time Protocol –
Nick Barendt, Kendall Correll, VXI Technology, Inc., Michael Branicky, Case Western Reserve University - (Paper). 93
(Presentation) 99
Investigation of IEEE 1588 on Gigabit Ethernet, Priority Tagged Frames and Ethernet Daisy Chain –
Sivaram Balasubramanian, Anatoly Moldovansky and Ken Harris, Rockwell Automation - (Paper). 111
(Presentation) 115
Determination of the IEEE 1588 Relevant Timing Behaviour of 100 Base-TX PHVs –
Christoph Thurnheer, J. Blatner, M. Rupf, and H. Weibel, Zurich University of Applied Sciences. 120
Update on High Precision Time Synchronization –
Dieter Vook, Bruce Hamilton, Andrew Fermandez, Jeff Burch, and Vamsi Srikantam, Agilent Technologies – (Paper). 127
(Presentation) 134

TECHNICAL PRESENTATION-SESSION II – IMPLEMENTATION TOPICS-2

Design of a FPGA-based Hardware IEEE 1588 Implementation –
John Guilford, Agilent Technologies - (Paper). 153
(Presentation) 163
IEEE 1588 Based Clock Synchronization for Non-Ethernet Networks –
George Gaderer, Hannes Muhr, Institute of Computer Technology, Vienna University of Technology, Thilo Sauter, Austrian Academy of Sciences, Nikolaus Kerö, Oregano Systems 175
Chip-Design Building Blocks for Precision Clock Synchronization in Ethernet Networks –
Nickolaus Kerö, Roland Holler, Oregano Systems, Hannes Muhr, George Gaderer, Institute of Computer Technology, Vienna University of Technology, Thilo Sauter, Austrian Academy of Sciences. 181
Experiences with IEEE 1588 in Higher Cascaded Ethernet Networks –
Dirk Mohl, Hirschmann Automation and Control. 192

TECHNICAL PRESENTATION-SESSION III – MEASUREMENT ISSUES AND APPLICATIONS

A Distributed Test and Measurement Application Using IEEE 1588 –
Tony Lanagan, National Instruments. 203
IEEE 1588 in Test and Measurement Applications as Specified in LXI Standard v1.0 –
Bob Rennard, Agilent Technologies 212
Application of IEEE 1588 in Substation Automation –
Lars Anderson, Alf Johansson, Christoph Brunner, ABB Switzerland Ltd., High Voltage Products 222

In Search of the Key to the Lock: Clock Synchronization Issues and Requirments in Semiconductor Manufacturing –
Ya-Shian Li, John Messina, National Institute of Standards and Technology – (Paper)..... 228
(Presentation)..... 240
An IEEE-1588 Grand Master Clock –
Doug Arnold, Harrel Huckeba, Chris Calley, Paul Skoog, Symmetricom. 250

TECHNICAL PRESENTATION-SESSION IV – GENERAL SESSION

Report on IEEE 1588 Standards Activity –
John C. Eidson, Agilent Technologies. 257
Plug-fest Introduction –
Dirk Mohl, Hirschmann Automation and Control..... 263

TECHNICAL PRESENTATION-SESSION V – TELECOMMUNICATIONS ISSUES AND APPLICATIONS

Using PTP for Synchronizing Legacy Networks –
Martin Burnicky, Udo Maltzahn, Heiko Gerstung, Meinberg Radio Clocks – (Paper)..... 267
(Presentation)..... 279
Implementation Considerations for IEEE 1588 in Telecom Applications –
George Zampetti, Steven Blackman, Vandana Upadhyay and Srini Bangalore, Symmetricom. 289
IEEE 1588 in Telecommunications and Field Trial Results–
Dave Tonks, Semtech. 297
Synchronizing IEEE 1588 Clocks Under the Presence of Significant Stochastic Network Delays –
C. Heitz, C. Iantosca, H. Weibel, Zurich University of Applied Sciences – (Paper)..... 309
(Presentation)..... 320

TECHNICAL PRESENTATION-SESSION VI – RESIDENTIAL ETHERNET

Precise Timing in a Residential Ethernet Environment –
Michael Johas Teener, Broadcom, Felix Feng and Eric Ryu, Samsung, Geoffrey M. Garner, Samsung Electronics. 329
Analysis of Clock Synchronization Approaches for Residential Ethernet –
Geoffrey M. Garner, Samsung Electronics – (Paper)..... 349
(Presentation)..... 368

TECHNICAL PRESENTATION-SESSION VII – INDUSTRIAL APPLICATIONS
Rotational Synchronization via IEEE 1588 –
Keen-Hun Leong, Agilent Technologies – (Paper) 388
(Presentation)..... 407
Application of IEEE 1588 to Synchronize Multiple Robot Controllers –
Michael Gerstenberger, Clemens Gmeiner and Stefan Mueller, Kuka Robotics – (Paper)..... 420
(Presentation)..... 430

TECHNICAL PRESENTATION-SESSION VIII – IMPLEMENTATION TOPICS-3

Migration Towards IEEE 1588 Time Synchronization Over Gigabit Ethernet –
Sven Nylund and Oyvind Holmeide, OnTime Networks – (Paper). 442
(Presentation)..... 444
Clock Synchronization Based on Transparent Clock Approach –
Matthias Wenk, Siemens – (Paper) 452
(Presentation)..... 460
Clock Phase Change Compensation Using Graham Scan –
Augusto Ciuffoletti, Universita' degli Studi di Pisa and Galina Antonova, General Electric. 467

A Simulation Study – Performance of IEEE 1588 Over Ethernet and IEEE 802.11b WLANs –
Dr. Nirmala Shenoy, Rochester Institute of Technology, John Fischer, Paul Myers,
Zeping Qui, Spectracom – (Paper). 477
(Presentation)..... 486
A Security Analysis of the IEEE 1588 Standard –
Jeanette Sin Mei Tsang, Konstantin Beznosov, Laboratory Education and Research in
Secure Systems Engineering (LERSSE) University of British Columbia. 497
IEE 1588 Security Extensions: Requirements and Proposed Solutions –
Stephan Schüler, Siemens Communications. 509

FINAL PARTICIPANTS LIST 527

AGENDA

Monday, October 10, 2005

- 8:15 - 9:00 AM: Plug-fest Registration.
- 9:00 - 5:00 PM: Plug-fest, moderator: Dirk Mohl, Hirschmann Automation and Control
  Plug-fest activities (Open only to those directly involved in plug-fest testing)
- 1:00 - 2:00 PM: Tutorial Registration
- 2:00 - 5:20 PM: **IEEE 1588 Tutorial, Tutorial Ouline-2005 Conference**
- 2:00 - 3:30 PM: **IEEE 1588 Basics**: John C. Eidson, Agilent Technologies
- 3:30 - 3:50 PM: Break
- 3:50 - 4:20 PM: **Telecommunications Applications**: Silvana Rodrigues, Zarlink Semiconductor
- 4:20 - 4:50 PM: **Industrial and Motion Control Applications**: Anatoly Moldovansky, Rockwell Automation; Ludwig Winkel, Siemens
- 4:50 - 5:20 PM: **Test and Measurement Applications**: John C. Eidson, Agilent Technologies

Tuesday, October 11, 2005

- 7:30 - 8:15AM: **Registration.**
- 8:15 - 8:40 AM: **Opening Session:**
  - **Opening comments and welcome**, Hans Weibel, Zurich University of Applied Sciences
  - **Welcome from the University,** Werner Inderbitzin, President of the Zurich University of Applied Sciences

Session 1:

(25 minute papers) **Implementation Topics-1**
Moderator: Dave Tonks, Semtech

- 8:40-9:05 AM: **Servo Design Considerations for Software-only Implementations of the Precision Time Protocol:** Nick Barendt, Kendall Correll, VXI Technology, Inc., Michael Branicky, Case Western Reserve University.
- 9:05-9:30 AM: **Investigation of IEEE 1588 on Gigabit Ethernet, Priority Tagged Frames and Ethernet Daisy Chain:** Sivaram Balasubramanian, Anatoly Moldovansky and Ken Harris, Rockwell Automation
- 9:30-9:55 AM: **Determination of the IEEE 1588 Relevant Timing Behaviour of 100Base-TX PHYs:** Christoph Thurnheer, J. Blattner, M. Rupf, and H. Weibel, Zurich University of Applied Sciences
- 9:55-10:20 AM: **Update on High Precision Time Synchronization:** Dieter Vook, Bruce Hamilton, Andrew Fernandez, Jeff Burch, and Vamsi Srikantam, Agilent Technologies
- 10:20-10:45 A: Break

Session 2:

(20 minute papers) **Implementation Topics-2:**
Moderator: Doug Arnold, Symmetricom

- 10:45-11:05 AM: **Design of a FPGA-based Hardware IEEE 1588 Implementation:** John Guilford, Agilent Technologies
- 11:05-11:25 AM: **IEEE-1588 Based Clock Synchronization for Non-Ethernet Networks:** George Gaderer, Hannes Muhr, Institute of Computer Technology, Vienna University of Technology, Thilo Sauter Austrian Academy of Sciences, Nikolaus Kerö, Oregano Systems
- 11:25-11:45 AM: **Chip-Design Building Blocks for Precision Clock Synchronization in Ethernet Networks:** Nikolaus Kerö, Roland Höller, Oregano Systems, Hannes Muhr, Georg Gaderer, Institute of Computer Technology, Vienna University of Technology, Thilo Sauter Austrian Academy of Sciences
- 11:45-12:05 AM: **Experiences with IEEE 1588 in Higher Cascaded Ethernet Networks:** Dirk Mohl, Hirschmann Automation and Control
- 12:05-1:00 PM: Lunch

Session 3:

(25 minute papers) **Measurement Issues and Applications:**
Moderator: Ludwig Winkel, Siemens

- 1:00-1:25 PM: **A Distributed Test and Measurement Application Using IEEE 1588**: Tony Lanagan, National Instruments
- 1:25-1:50 PM: **IEEE 1588 in Test and Measurement Applications as Specified in LXI Standard v1.0:** Bob Rennard, Agilent Technologies.
- 1:50-2:15 PM: **Application of IEEE 1588 in Substation Automation:** Lars Andersson, Alf Johansson, Christoph Brunner ABB Switzerland Ltd., High Voltage Products
- 2:15-2:40 PM: **In Search of the Key to the Lock: Clock Synchronization Issues and Requirements in Semiconductor Manufacturing:** Ya-Shian Li, John Messina, National Institute of Standards and Technology
- 2:40-3:05 PM: **An IEEE-1588 Grand Master Clock:** Doug Arnold, Paul Skoog, Symmetricom

Session 4:

General Session: Moderator: Kang Lee, NIST

- 3:05-3:15 PM: **Report on IEEE 1588 Standards Activity:** John C. Eidson, Agilent Technologies
- 3:15-3:25 PM: **Plug-fest Introduction:** Dirk Mohl, Hirschmann Automation and Control
- 3:25-5:30 PM: **Combined plug-fest/demo viewing and afternoon break**
- 7:30 PM: Conference Dinner

Wednesday, October 12, 2005

- 7:30-8:15 AM: **Registration.**

Session 5:
(20/25 minute papers) **Telecommunications Issues and Applications:**
Moderator: Matthias Wenk, Siemens

- 8:15-8:35 AM: **Using PTP for Synchronizing Legacy Networks:** Heiko Gerstung, Meinberg Radio Clocks
- 8:35-8:55 AM: **Network Synchronization Overview and Analysis of Current Technologies:** Francesco Caggioni, TranSwitch Corp,
- 8:55-9:15 AM: **Implementation Considerations for IEEE 1588 in Telecom Applications:** Steven Blackman, Vandana Upadhyay and Srini Bangalore, Symmetricom,
- 9:15-9:35 AM: **IEEE 1588 in Telecommunications and Field Trial Results:** Dave Tonks, Semtech,
- 9:35-10:00 AM: **Synchronizing IEEE 1588 Clocks Under the Presence of Significant Stochastic Network Delays:** C. Heitz, C. Iantosca, H. Weibel, Zurich University of Applied Sciences
- 10:00-10:25 AM: Break

Session 6:
(25 minute papers) **Residential Ethernet:**
Moderator: John C. Eidson, Agilent Technologies

- 10:25-10:50 AM: **Precise Timing in a Residential Ethernet Environment:** Michael Johas Teener, Broadcom, Felix Feng and Eric Ryu, Samsung.
- 10:50-11:15 AM: **Analysis of Clock Synchronization Approaches for Residential Ethernet:** Geoffrey M. Garner, Samsung Electronics

Session 7:
(20/25 minute papers) **Industrial Applications:**
Moderator: Ken Harris, Rockwell Automation

- 11:15-11:35 AM: **Rotational Synchronization via IEEE 1588:** Keen-Hun Leong, Agilent Technologies
- 11:35-12:00 PM: **Application of IEEE 1588 to Synchronize Multiple Robot Controllers:** Michael Gerstenberger, Clemens Gmeiner and Stefan Mueller, Kuka Robotics
- 12:00-1:00 PM: Lunch

Session 8:
(25 minute papers) **Implementation Topics-3:**
Moderator: Hans Weibel, Zurich University of Applied Sciences

- 1:00-1:25 PM: **Migration Towards IEEE 1588 Time Synchronization Over Gigabit Ethernet:** Øyvind Holmeide and Sven Nylund, OnTime Networks
- 1:25-1:50 PM: **Clock Synchronization Based on Transparent Clock Approach:** Matthias Wenk, Siemens
- 1:50-2:15 PM: **Clock Phase Change Compensation Using Graham Scan:** August Ciuffoletti, Universita' degli Studi di Pisa, and Galina Antonova, General Electric
- 2:15-2:40 PM: **A Simulation Study - Performance of IEEE 1588 Over Ethernet and IEEE 802.11b WLANs:** Dr. Nirmala Shenoy, Rochester Institute of Technology, John Fischer, Paul Myers, Zeping Qui, Spectracom
- 2:40-3:05 PM: Break
- 3:05-3:30 PM: **A Security Analysis of the IEEE 1588 Standard:** Jeanette Sin Mei Tsang, Konstantin Beznosov, Laboratory Education and Research in Secure Systems Engineering (LERSSE) University of British Columbia
- 3:30-3:55 PM: **IEEE 1588 Security Extensions: Requirements and Proposed Solutions:** Stephan Schüler, Siemens Communications
- 3:55-4:15 PM: **Closing Session:** Kang Lee, National Institute of Standards and Technology

EXECUTIVE SUMMARY

The IEEE 1588 standard defines a protocol to synchronize real-time clocks in the nodes of a distributed system that communicate using a network. The 3rd IEEE 1588 conference was hosted by the Zurich University of Applied Sciences in Winterthur, Switzerland on October 10-12, 2005. The objective of the conference is to provide a forum for reporting on technical and standards issues, product development, and implementation experiences using the IEEE 1588 standards. The key findings of the conference are that various industry sectors view IEEE 1588 as an important emerging standard for precision clock synchronization and some sectors of industry such as residential Ethernet, telecommunications, and test and measurement have shown a need for higher precision in clock synchronization than specified in the current version of the IEEE 1588-2002 standard. The recommendation is to advance the standard to meet the most stringent industry requirements.

The Institute of Electrical and Electronics Engineers (IEEE), the Instrumentation and Measurement Society, and the National Institute of Standards and Technology are technical cosponsors of the conference. Werner Inderbitzin, President of the Zurich University of Applied Sciences, opened the conference with a warm welcome to the university and to Zurich. More than 80 international attendees, coming from diverse application areas such as instrumentation and measurement, industrial automation, aerospace, power generation, semiconductor manufacturing, and telecommunications participated in the conference.

The three-day event began with a well attended tutorial on the IEEE 1588 standard and its applications in various sectors of industry. A key event of this conference was a plug-fest. At the plug-fest, the interoperability of IEEE 1588-based components and devices was demonstrated by thirteen participating organizations, up from seven last year. The plug-fest was organized and chaired by Dirk Mohl of Hirschmann Automation and Control. Participating organizations included Agilent Technologies, Hirschmann Automation and Control GmbH, IXXAT Automation GmbH, KUKA Controls GmbH, Meinberg Funkuhren, National Instruments, Resolute Networks Inc., Rockwell Automation, Semtech, Symmetricom Inc., Westermo OnTime AS, VXI Technology, and the Zurich University of Applied Sciences. In addition to testing basic synchronization, this year's plug-fest tested the operation of the best master clock algorithm and the application of management messages. The tests demonstrated the interoperability of the implementations and synchronization to levels of 40 ns for implementations with hardware assist.

During the general session, the conference participants viewed the interoperability demonstration of plug-fest devices which included three grandmaster clock devices linked to the global positioning system (GPS). Boundary clocks and several end devices were exhibited. In addition to the devices that were part of the plug-fest, several other devices were demonstrated including an IEEE 1588 implementation on a gigabit Ethernet, an end-to-end transparent clock prototype, and a motion control system based on IEEE 1588.

The technical sessions covered diverse subjects such as primary timing reference sources for IEEE 1588 systems, nanosecond-level clock synchronization, network simulation environment for clock synchronization, physical chip issues, security, synchronizing legacy networks, synchronization approach for resident Ethernet, and implementation on a gigabit Ethernet. Furthermore, there were a number of application papers covering the areas of industrial automation, military, test and measurement, power generation, and telecommunications.

Based on the presentations, plug-fest interaction, and discussions at the conference, the IEEE 1588 standard is very well-received by various sectors of industry. As indicated by the plug-fest participants, IEEE 1588-based products are now available and more are in the development phase. These products can be used as components for applications requiring precision clock synchronization.

This report contains presentation slides and papers contributed by participants of the conference. The slides will also be posted on the IEEE 1588 website at http://ieee1588.nist.gov.

Conference Co-chairs
John C. Eidson, Agilent Technologies
Kang Lee, National Institute of Standards and Technology
Hans Weibel, Zurich University of Applied Sciences

Plug-fest Chair
Dirk Mohl, Hirschmann Automation and Control

PRESENTATIONS

IEEE-1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

-A Tutorial-

John Eidson
October 10, 2005
john_eidson@agilent.com

Agilent Technologies

© Copyright 2005 Agilent Technologies, Inc

Outline

1. General overview of the technology and applications
2. Guide to the standard- a detailed analysis of the major clauses
3. IEEE 1588 interoperability/conformance topics
4. Implementation topics
5. Applications
   - Industrial automation- Anatoly Moldovansky, Rockwell; Ludwig Winkel, Siemens
   - Telecommunications- Silvana Rodrigues, Zarlink
   - Test & Measurement- John Eidson, Agilent

General Overview of the Technology

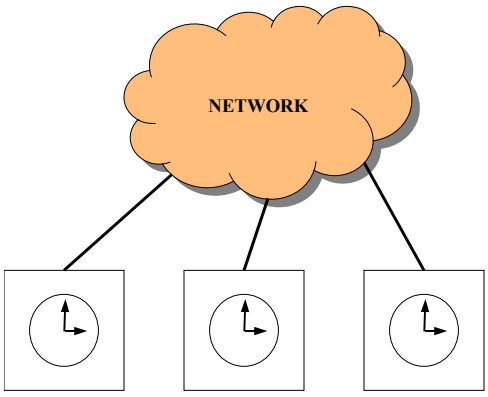
a. **Purpose**

b. **Status and activities surrounding IEEE 1588**

c. **Comparison to other protocols**

The Purpose of IEEE 1588

IEEE 1588 is a protocol designed to synchronize real-time clocks in the nodes of a distributed system that communicate using a network.



The Status of IEEE 1588

- **Approved by the IEEE-SA Review Committee on September 12, 2002**
- **Published as IEEE 1588-2002 on November 8, 2002**
- **Available from the IEEE http://standards.ieee.org**
- **Approved as IEC standard IEC 61588 on May 21, 2004**
- **Products and installations started appearing in late 2003**
- **Conferences on IEEE 1588 held in 2003, 2004, 2005**
- **P1588 committee in process of extending the standard-target completion in late 2006**
- **Current information may be found at http://ieee1588.nist.gov**

Comparison to Other Protocols

	IEEE-1588	NTP	GPS	TTP	SERCOS
Spatial extent	A few subnets	Wide area	Wide area	Local bus	Local bus
Communi-cations	Network	Internet	Satellite	Bus or star	Bus
Target accuracy	Sub-microsecond	Few milliseconds	Sub-microsecond	Sub-microsecond	Sub-microsecond
Style	Master/slave	Peer ensemble	Client/server	Distributed	Master/Slave
Resources	Small network message and computation footprint	Moderate network and computation footprint	Moderate computation footprint	Moderate	Moderate

Comparison to Other Protocols (continued)

	IEEE 1588	NTP	GPS	TTP	SERCOS
Latency correction	Yes	Yes	Yes	Configured	No
Protocol specifies security	No (V2 may include security)	Yes	No	No	No
Administration	Self organizing	Configured	N/A	Configured	Configured
Hardware?	For highest accuracy	No	RF receiver and processor	Yes	Yes
Update interval	~2 seconds	Varies, nominally seconds	~1 second	Every TDMA cycle, ~ms	Every TDMA cycle, ~ms

Comparison to Other Protocols (summary)

IEEE 1588: Target is groups of relatively stable components, locally networked (a few subnets), cooperating on a set of well defined tasks.

NTP: (Network Time Protocol, RFC 1305). Target is autonomous systems widely dispersed on the Internet.

GPS: (Satellite based Global Positioning System of the US Department of Defense): Target is autonomous, widely dispersed systems.

TTP(www.ttpforum.org), SERCOS (IEC 61491): Target is tightly integrated, usually bus or specialized TDMA network based closed systems.

Guide to the Standard-
(A detailed analysis of the major clauses of version 1)

a. **Overview and goals of the standard**
b. **Synchronization messages and methodology**
c. **Selection of master clocks**
d. **State machine and events**
e. **Timing considerations**
f. **Management messages**

Objectives of IEEE 1588

- **Sub-microsecond synchronization of real-time clocks in components of a networked distributed measurement and control system***
- **Intended for relatively localized systems typical of industrial automation and test and measurement environments. ***
- **Applicable to local area networks supporting multicast communications (including but not limited to Ethernet)**

***indicates objectives that may be extended in version 2**

Objectives of IEEE 1588 (continued)

- Simple, administration free installation
- Support heterogeneous systems of clocks with varying precision, resolution and stability
- Minimal resource requirements on networks and host components.

The IEEE 1588 Standard Defines:

- Descriptors characterizing a clock
- The states of a clock and the allowed state transitions
- IEEE 1588 network messages, fields, and semantics
- Datasets maintained by each clock
- Actions and timing for all IEEE 1588 network and internal events

- **Critical physical specifications**
- **A suite of messages for monitoring the system**
- **Specifications for an Ethernet based implementation**
- **Conformance requirements**
- **Implementation suggestions**

Overview of the IEEE 1588 Standard

Clause	Purpose	Annex	Purpose
1	Scope	**A**	User Information
2	Standards References	**B**	Time Scales
3	Definitions	**C**	Subdomain Maps
4	Notation Convention	**D**	Ethernet UDP/IP Implementation
5	Datatypes	**E**	Bibliography
6	Protocol Overview		
7	Protocol		
8	Message Specifications		
9	Conformance		

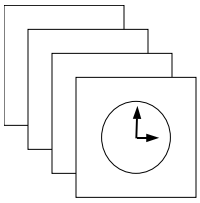
WARNING

The IEEE has rather strict rules on interpreting IEEE Standards. No individual or organization can issue official interpretations or provide definitive answers to questions of interpretation. This must be done by an IEEE authorized committee. Even this committee cannot extend, correct, or change the standard- this must be done by ballot.

However-We can learn from and share our collective experience.

Clause 6: PTP Clock Synchronization Model

QUESTION: How do we take a collection of clocks, message types, clock properties, networks, etc. and produce a consistent time base in all the participating clocks?



MESSAGE TYPES	CLOCK PROPERTIES
Sync	UUID
Delay_Req	Stratum
Follow_Up	Identifier
Delay_Resp	State
Management	Variance ….

NETWORK COMMUNICATION FORMS:
Ethernet (UDP/IP), DeviceNet, L2 Ethernet, 802.11b, …

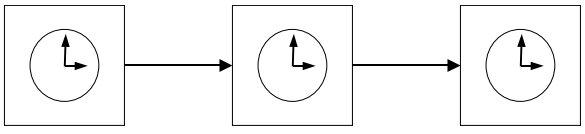
Guide to the Standard-
(A detailed analysis of the major clauses)

a. **Overview and goals of the standard**
b. **Synchronization messages and methodology**
c. **Selection of master clocks**
d. **State machine and events**
e. **Timing considerations**
f. **Management messages**
g. **Time scales**
h. **Annex D**

Clause 6: IEEE 1588 Synchronization Basics

Step 1: Organize the clocks into a master-slave hierarchy (based on observing the clock property information contained in multicast Sync messages)

Step 2: Each slave synchronizes to its master (based on Sync, Delay_Req, Follow_Up, and Delay_Resp messages exchanged between master and its slave)

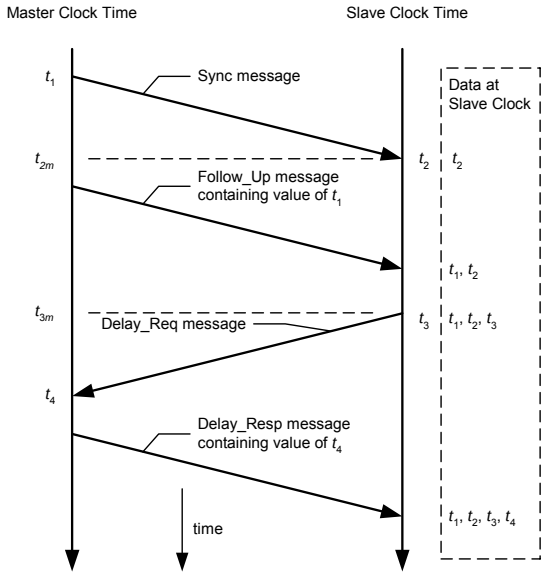


Grandmaster Clock
This clock determines the time base for the system

Slave to the Grandmaster Clock and Master to its Slave

Slave to its Master

Clause 6: Synchronization Basics (continued)



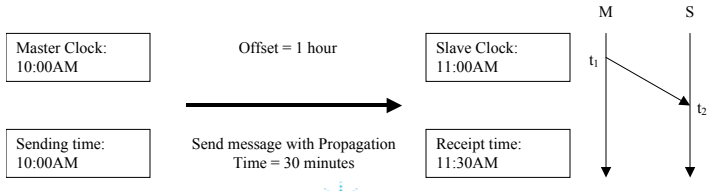
Clause 6: Synchronization Basics (continued)

To synchronize a pair of clocks, First:

- **Send a message, (Sync message), from master to slave and measure the apparent time difference between the two clocks. MS_difference = slave's receipt time – master's sending time**

  $= t_2 - t_1$

- **MS_difference = offset + MS delay (by inspection)**
- **For example:**
  MS_difference = slave's receipt time – master's sending time
  90 minutes = 11:30 – 10:00



Clause 6: Synchronization Basics (continued)

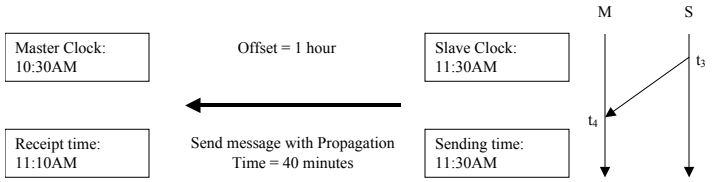
Second:

- **Send a message, (Delay_Req message), from slave to master and measure the apparent time difference between the two clocks.**
  SM_difference = master's receipt time – slave's sending time

  $= t_4 - t_3$

SM_difference = – offset + SM delay (by inspection)

- **For example:**
  SM_difference = master's receipt time – slave's sending time – 20 minutes = 11:10 – 11:30



Clause 6: Synchronization Basics (continued)

The result is that we have the following two equations:

MS_difference = offset + MS delay

SM_difference = – offset + SM delay

With two measured quantities:

MS_difference = 90 minutes

SM_difference = – 20 minutes

And three unknowns:

offset , MS delay, and SM delay

Clause 6: Synchronization Basics (continued)

Rearranging the two equations:

MS_difference = offset + MS delay

SM_difference = – offset + SM delay

We get:

offset = {(MS_difference – SM_difference) – (MS delay – SM delay)}/2

MS delay + SM delay = {MS_difference + SM_difference}

ASSUME: MS delay = SM delay = one_way_delay

Then:

offset = {MS_difference – SM_difference}/2

one_way_delay = {MS_difference + SM_difference}/2

Clause 6: Synchronization Basics (continued)

offset = {MS_difference – SM_difference}/2

one_way_delay = {MS_difference + SM_difference}/2

In our example using the two measured quantities:

MS_difference = 90 minutes

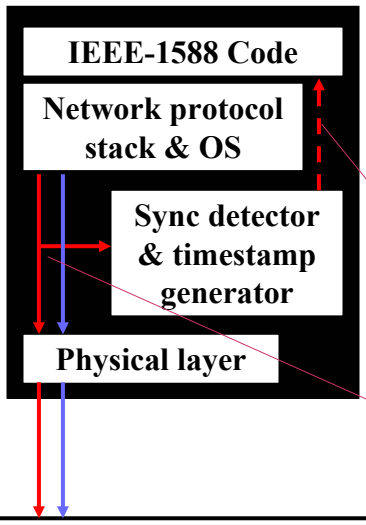
SM_difference = – 20 minutes

We get:

offset = {90 – (– 20)}/2 = 55 minutes (not actual 60)

one_way_delay = {90 + (– 20)}/2 = 35 minutes (not 30 or 40)

Synchronization Details (clauses 6 & 7)



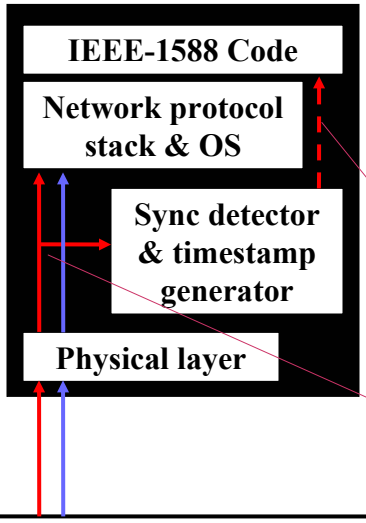
Master clock sends:

1. Sync message
2. Follow_up message

Time at which a Sync message passed the Timestamp Point (t_1)

Timestamp Point

Synchronization Details (continued)



Slave clock receives:

1. Sync message
2. Follow_up message

Time at which a Sync message passed the Timestamp Point (t_2)

Timestamp Point

Synchronization Details (continued)

Sync messages:

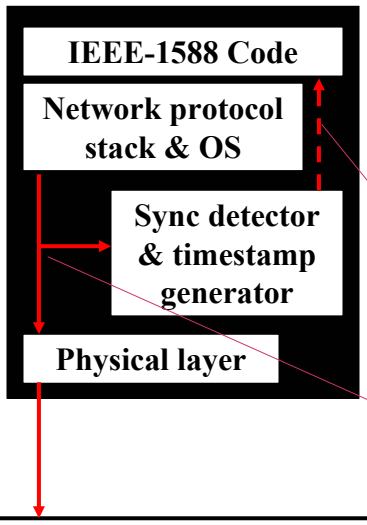
- **Issued by clocks in the 'Master' state**
- **Contain clock characterization information**
- **Contain an estimate of the sending time ($\sim t_1$)**
- **When received by a slave clock the receipt time is noted**
- **Can be distinguished from other legal messages on the network**
- **For best accuracy these messages can be easily identified and detected at or near the physical layer and the precise sending (or receipt) time recorded**

Synchronization Details (continued)

Follow_Up messages:

- **Issued by clocks in the 'Master' state**
- **Always associated with the preceding Sync message**
- **Contain the 'precise sending time= (t_1)' as measured as close as possible to the physical layer of the network**
- **When received by a slave clock the 'precise sending time' is used in computations rather than the estimated sending time contained in the Sync message**

Synchronization Details (continued)



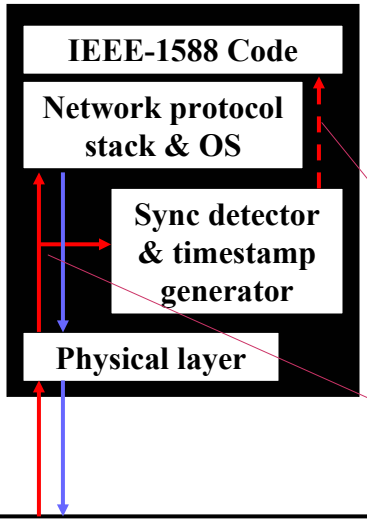
Slave clock sends:

- Delay_Req message

Time at which a Delay_Req message passed the Timestamp Point (t_3)

Timestamp Point

Synchronization Details (continued)



Master clock receives:

- Delay_Req message

Master clock sends:

- Delay_Resp message

Time at which a Delay_Req message passed the Timestamp Point (t_4)

Timestamp Point

Delay_Req messages:

- **Issued by clocks in the 'Slave' state**
- **The slave measures and records the sending time** (t_3)
- **When received by the master clock the receipt time is noted** (t_4)
- **Can be distinguished from other legal messages on the network**
- **For best accuracy these messages can be easily identified and detected at or near the physical layer and the precise sending (or receipt) time recorded**

Delay_Resp messages:

- **Issued by clocks in the 'Master' state**
- **Always associated with a preceding Delay_Req message from a specific slave clock**
- **Contain the receipt time of the associated Delay_Req message** (t_4)
- **When received by a slave clock the receipt time is noted and used in conjunction with the sending time of the associated Delay_Req message as part of the latency calculation**

Synchronization Details (continued)

Synchronization computation (in the Slave clock):

offset = receipt time – precise sending time – one way delay (for a Sync message)

one way delay = {master to slave delay + slave to master delay}/2 (assumes symmetric delay)

master to slave delay = receipt time – precise sending time (for a Sync message)

slave to master delay = Delay_Req receipt time -precise sending time (of a Delay_Req message)

From this offset the slave corrects its local clock!

From This Offset the Slave Corrects its Local Clock!

BUT: The standard says nothing about how to do this.

(more later)

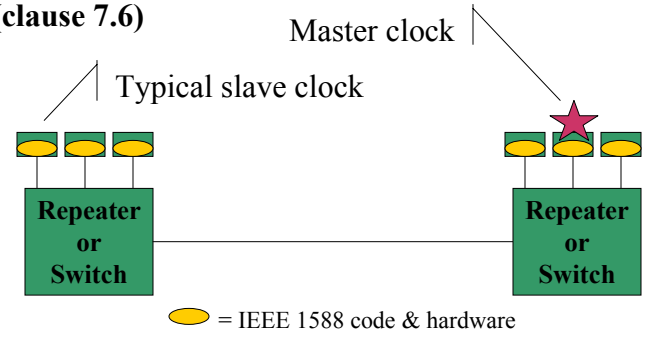
Guide to the Standard-
(A detailed analysis of the major clauses)

a. **Overview and goals of the standard**
b. **Synchronization messages and methodology**
c. **Selection of master clocks**
d. **State machine and events**
e. **Timing considerations**
f. **Management messages**
g. **Time scales**
h. **Annex D**

Selecting a Master Clock-Single Subnet

Self-configuring based on clock characteristics and network topology

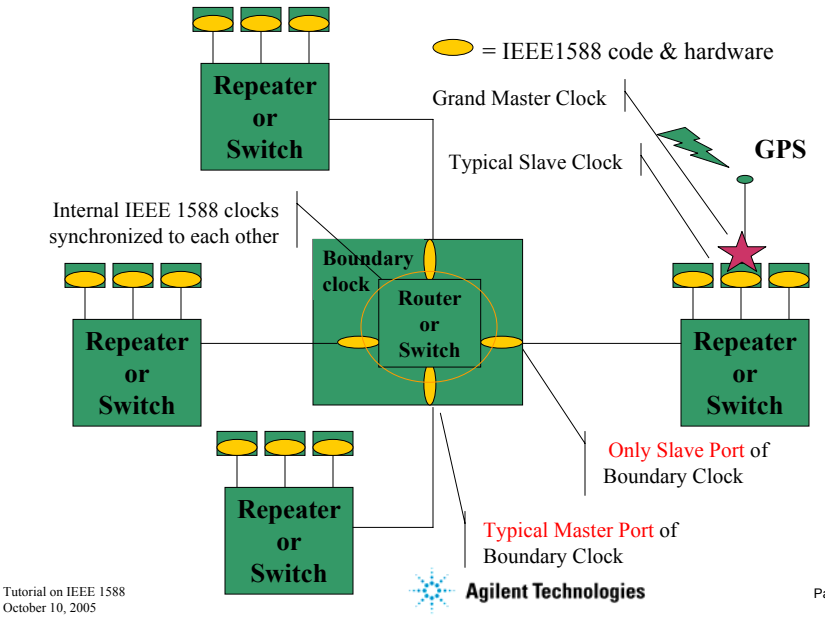
- **Based on information contained in 'Sync' messages**
- **All clocks run an identical 'Best Master Clock' algorithm (clause 7.6)**



Selecting a Master Clock-Simplified (clause 7.6.1)

- **A clock at startup listens for a time SYNC_RECEIPT_TIMEOUT**
- **A master clock (clock in the PTP_MASTER state) issues periodic Sync messages (period is called the sync_interval)**
- **A master clock may receive Sync messages from other clocks (who for the moment think they are master) which it calls 'foreign masters'**
- **Each master clock uses the Best Master Clock algorithm to determine whether it should remain master or yield to a foreign master.**
- **Each non-master clock uses the Best Master Clock algorithm to determine whether it should become a master.**

IEEE 1588 Multiple Subnet Topology



Multiple Subnet Synchronization & Master Clock Selection (more details)

- **Boundary clocks do NOT pass Sync, Follow_Up, Delay_Req, or Delay_Resp messages. Boundary clocks thus segment the network as far as IEEE 1588 synchronization is concerned.**
- **Within a subnet a port of a boundary clock acts just like an ordinary clock with respect to synchronization and best master clock algorithm**
- **The boundary clock internally selects the port that sees the 'best clock' as the single slave port. This port is a slave in the selected subnet. All other ports of the boundary clock internally synchronize to this slave port.**

- **Boundary clocks define a parent-child hierarchy of master-slave clocks.**
- **The best clock in the system is the Grand Master clock.**
- **If there are cyclic paths in the network topology the best master clock algorithm reduces the logical topology to an acyclic graph.**

Best Master Clock Algorithm-overview (clause 7.6)

1. A master clock 'A' can receive Sync messages from other potential master clocks- 'B', 'C',…
2. Clock 'A' decides:
   a. Which of the clocks 'B', 'C' ,… is the 'best' clock
   b. Whether clock 'A' is better than the best of 'B', 'C' ,…
3. Using the Best Master Clock algorithm, BMC, it does this by pair wise comparisons of the data sets describing each of the clocks.
4. Based on the results of this comparison the BMC returns a recommended clock state: in simple situations either master or slave.
5. All clocks operate on the same information and therefore arrive at consistent results.
6. Data for these comparisons logically is maintained by each clock in one of several data sets.

Datasets Maintained by Each Clock (clause 7.4)

The following are PER CLOCK data sets:

•Default data set: Properties of the local clock that determine its behavior and performance when it is the grandmaster clock

•Global time properties data set: Time base properties

•Current data set: Current synchronization and topological operational properties

The following are PER CLOCK PORT data sets:

•Parent data set: Properties of the parent and grandmaster

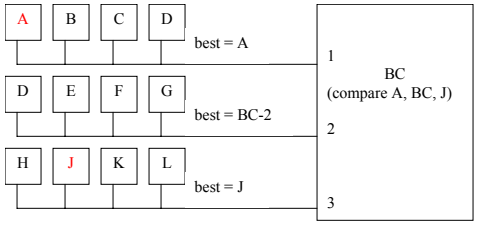
•Port configuration data set: Clock port properties

•Foreign master data set: Identification of Sync messages from potential master clocks-part of a qualification scheme to reduce thrashing

"It does this by pair wise comparisons of the data sets…"

•On a subnet an ordinary clock sees itself and others on the same subnet: default and foreign master data sets: e.g. 'A' sees B,C,D, and BC port 1

•A boundary clock sees itself and all clocks on its several subnets: default and the foreign master set for each port: e.g. BC sees all ordinary clocks, A,…L, plus itself.



IEEE 1588 Characterization of Clocks (clause 6)

The following are the principal items used by the BMC.

- **Based on primary source of time, e.g. GPS, local oscillator…**
- **Accuracy**
- **Variance**
- **Preferred set membership**
- **Type: Boundary clock (spans subnets) or ordinary clock**
- **UUID**

Best Master Clock Algorithm-details clause 7.6

The BMC algorithm consist of two sub algorithms:

1. **State decision algorithm: using the results of comparisons of all pairs of relevant data sets this produces a recommended state.**
2. **Data set comparison algorithm: a binary relation using specific information from the data sets of the two clock ports being compared:**
   a. **Select the clock that derives its time from the better grandmaster**
   b. **If the grandmasters are equivalent choose the 'closest' grandmaster**
   c. **If the above fail to indicate a choice use tie-breaking (UUID)**

Data Set Comparison Algorithm (clause 7.6.4)

Standard has a big flow chart (figures 17, 18, 19 & 20) and a table (20) defining this algorithm. The net effect is to define a hierarchy of choices of which the first one satisfied determines which of the two data sets represents the 'better' clock (port). The hierarchy is:

1.Preferred: (designates a set from which GM is selected)

2.Stratum: (clause 6.2.4.3- primary or secondary standard)

3.Identifier: (6.2.4.5- accuracy of clock's time base)

4.Variance: (6.2.4.8- stability and noise of clock)

5.'Closest': minimum spanning tree algorithm (key to understanding mechanism is Table 21)

6.UUID (tiebreaker)

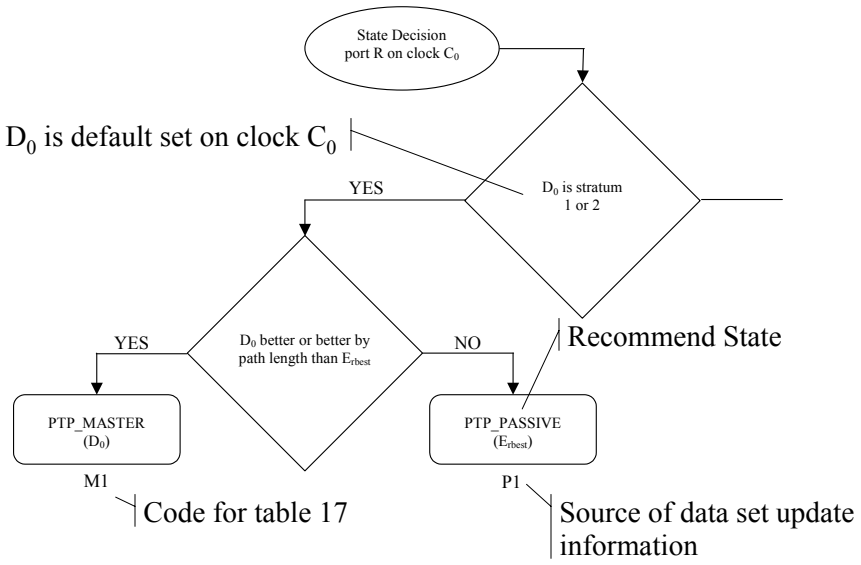
State Decision Algorithm (clause 7.6.3 & figure 16)

The result of this algorithm is:

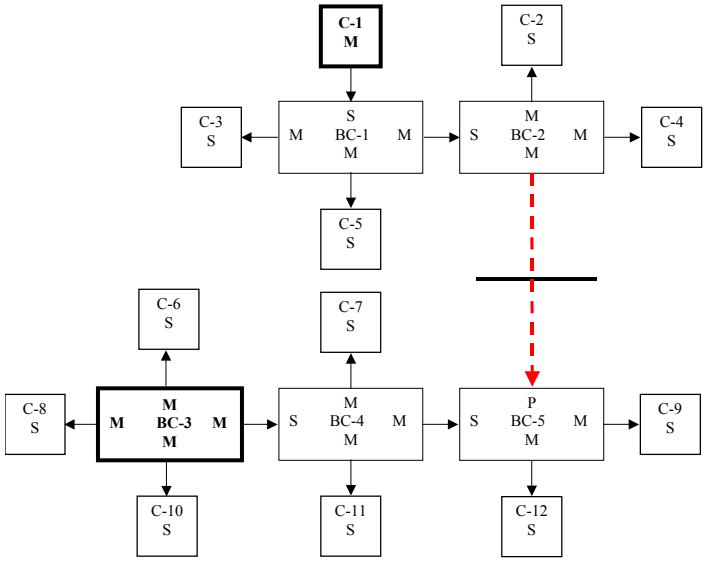
- **A 'recommended state': drives the state machine of 7.3**
- **Update specification for data sets**

A CAREFUL study of figure 16 reveals all. For example:

State Decision Algorithm (clause 7.6.3 & figure 16)



The Result of the State Decision Algorithm:



Guide to the Standard-
(A detailed analysis of the major clauses)

a. **Overview and goals of the standard**
b. **Synchronization messages and methodology**
c. **Selection of master clocks**
d. **State machine and events**
e. **Timing considerations**
f. **Management messages**
g. **Time scales**
h. **Annex D**

State Machine (clause 7.3)

There are 9 states defined in IEEE 1588:

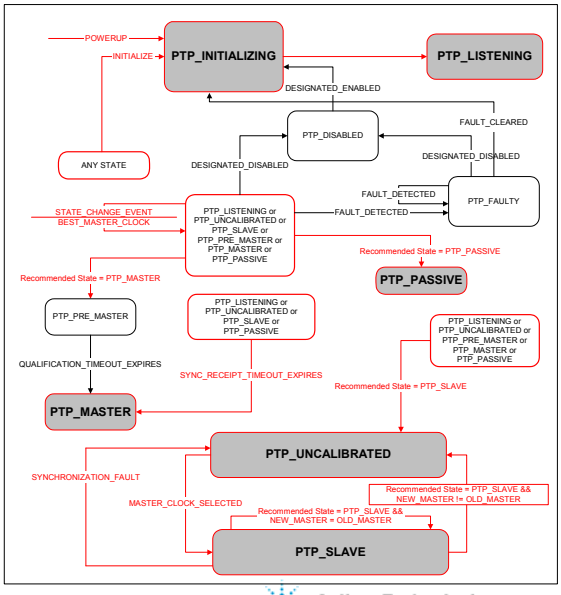
1. **PTP_INITIALIZING: Initialization- data sets, hardware**
2. **PTP_FAULTY: fault state**
3. **PTP_DISABLED: allows removal of a clock**
4. **PTP_LISTENING: orderly addition of clocks to net**
5. **PTP_PRE_MASTER: transitions in complex topologies**
6. **PTP_MASTER: clock is source of time to its slaves**
7. **PTP_PASSIVE: used to segment network**
8. **PTP_UNCALIBRATED: transition state to slave**
9. **PTP_SLAVE: synchronizing to it's master**

State Machine Events (clause 7.5)

There are several events that MAY lead to a state change:

1. **Initialization**
2. **Receipt of any message**
3. **STATE_CHANGE_EVENT: clause 7.5.8 (at least once/sync interval !)**
4. **Transmission of a message**
5. **SYNC_RECEIPT_TIMEOUT_EXPIRES**
6. **Sync interval timeout expires**
7. **QUALIFICATION_TIMEOUT_EXPIRES**
8. **BMC completes**
9. **Detection of an internal fault**
10. **Synchronization changes in a local clock**
11. **Events related to an external timing signal**

State Machine (simplified, clause 7.3, fig 9)



Guide to the Standard-
(A detailed analysis of the major clauses)

a. **Overview and goals of the standard**
b. **Synchronization messages and methodology**
c. **Selection of master clocks**
d. **State machine and events**
e. **Timing considerations**
f. **Management messages**
g. **Time scales**
h. **Annex D**

Timing Considerations (clause 7.11)

- **IEEE 1588 timing is centered around the sync interval**
- **Clause 7.11 specifies the rates at which events and messages must be processed by the local clock**
- **The most complex specification deals with how often slave clocks issue Delay_Req messages:**
  - **Randomized to reduce network and master clock processing loads**
  - **Randomization is first over multiple sync intervals and second within the selected interval.**

Guide to the Standard-
(A detailed analysis of the major clauses)

a. **Overview and goals of the standard**
b. **Synchronization messages and methodology**
c. **Selection of master clocks**
d. **State machine and events**
e. **Timing considerations**
f. **Management messages**
g. **Time scales**
h. **Annex D**

Management Messages (clause 7.12 & 6.2.2.1)

- **Management messages provide external visibility to the several data sets maintained within each clock**
- **Management messages provide a mechanism to modify certain parameters within these data sets, e.g. sync_interval, subdomain_name**
- **Management messages provide a mechanism to drive certain state changes. For example initialization, disabling, setting the time in the grand master, … can be forced using a management message.**

Guide to the Standard-
(A detailed analysis of the major clauses)

a. **Overview and goals of the standard**
b. **Synchronization messages and methodology**
c. **Selection of master clocks**
d. **State machine and events**
e. **Timing considerations**
f. **Management messages**
g. **Time scales**
h. **Annex D**

IEEE 1588 Time Scales (Annex B)

- **The time base in an IEEE 1588 system is the time base of the Grandmaster Clock. The epoch and rate is determined by the grandmaster.**
- **All other clocks synchronize (perhaps via boundary clocks) to the grand master.**
- **The Grandmaster Clock time base is implementation and application dependent.**
- **If the Grandmaster Clock maintains a UTC time base, the IEEE 1588 protocol distributes leap second information to the slaves if it is available.**

IEEE 1588 Time Scales (Annex B): EPOCH

Time Base	IEEE 1588 User Defined	IEEE 1588 UTC	GPS UTC	NTP UTC
Epoch	User Defined	0:00:00 1 January 1970	0:00:00 6 January 1980	0:00:00 1 January 1900
Rollover Frequency	~$9x10^6$ years	~$9x10^6$ years	1024 weeks (~2019)	136 years (~2036)
Linear Time Base	Yes	Yes (offset TAI)	Yes (offset TAI)	No (leap second discontinuity)
Civil Calendar Events	Hard leap seconds	Hard leap seconds	Hard leap seconds	Easy
Duration & Relative Events	Easy	Easy	Easy	Hard leap seconds

Guide to the Standard-
(A detailed analysis of the major clauses)

a. Overview and goals of the standard
b. Synchronization messages and methodology
c. Selection of master clocks
d. State machine and events
e. Timing considerations
f. Management messages
g. Time scales
h. Annex D

ANNEX D: IEEE 1588 on UDP/IP ETHERNET

1. Defines the message time stamp point: The start of the first bit of the octet following the start of frame delimiter
2. Defines relevant fields in Ethernet header
3. Defines the mapping of clause 8 messages onto Ethernet frame user space
4. Defines IEEE 1588 uuid_field when using Ethernet: Based on Ethernet MAC address

ANNEX D: IEEE 1588 on UDP/IP ETHERNET (continued)

5. Defines IEEE 1588 addressing when using UDP/IP on Ethernet

Port category	IANA Name	Value
EventPort	Ptp-event	319
GeneralPort	Ptp-general	320

Address name	IANA Name	Value
DefaultPTPdomain	PTP-primary	224.0.1.129
AlternatePTPdomain1	PTP-alternate1	224.0.1.130
AlternatePTPdomain2	PTP-alternate2	224.0.1.131
AlternatePTPdomain3	PTP-alternate3	224.0.1.132

AND!

Everything covered so far exists within a scope.

The scope is defined by the value of the subdomain_name parameter of the default data set. (clauses 6.2.5 & 7.4.2)

All activity such as messages, time base, state machines, etc. in one subdomain is completely independent of similar activity in another subdomain, even on the same network medium.

IEEE 1588 Interoperability/Conformance Topics

1. Interoperability and conformance are NOT the same thing!
2. Clause 9 defines three levels of clock conformance and a minimal set of system conformance requirements.
3. Individual clock conformance:
   a. Fully conformant: meets all aspects of IEEE 1588 standard
   b. Slave only: Always defers to E_{best} (clause 7.6) as selected by BMC algorithm. Never issues Sync, Delay_Resp, or Follow_Up messages
   c. Management only: Only issues management messages.
4. System: Conformant clocks, One fully conformant clock, common system parameters, no non-specified transport links

IEEE 1588 Interoperability/Conformance Topics

IEEE 1588 network messages represent the critical interface to an IEEE 1588 clock port.

Detailed network independent specifications on the fields, meanings, data types, etc. for each of the 5 defined IEEE 1588 messages are given in clause 8.

Specific mappings of the message specifications onto a particular network transport are defined in Annexes to the standard.

Currently the only such mapping is to UDP/IP on Ethernet.

Implementation Topics

1. Minimal implementations
2. Accuracy issues
3. Application level support

Minimal Implementations

IEEE 1588 specifies very few optional features:

- Slave only nodes (conformance clause 9.2.2)
- Follow_Up capable (clause 6.2.4.6). This is tied to the issue of hardware assist in time stamping Sync and Delay_Req messages.
- External timing signal (clause 7.5.20)
- Burst mode (clause 7.5.5, 7.5.9, 7.5.11)
- Parent statistics (clause 7.4.4.8)

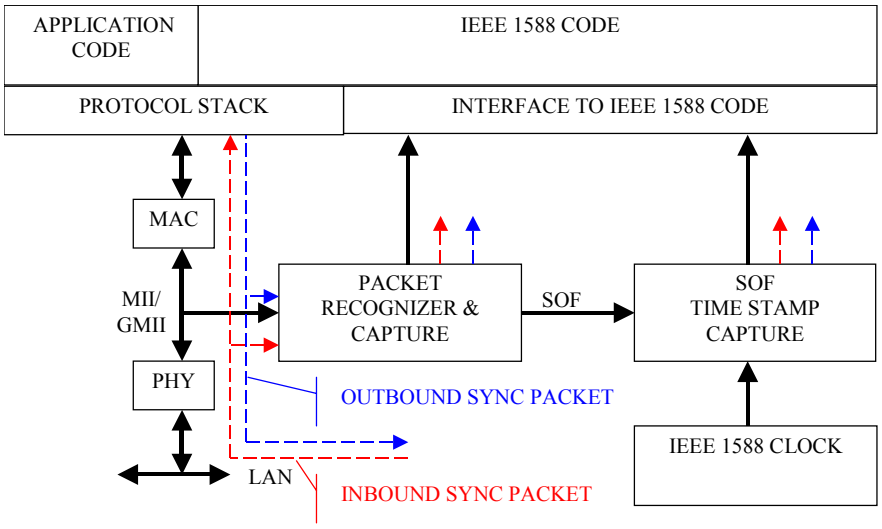
Minimal Implementations (Follow_Up capable)

1. Clocks generate a time stamp when a Sync message is sent or received.
2. Can be done in hardware (e.g. at MII and is the most accurate), ISR or kernel level, or at application level (least accurate)
3. Can be communicated:
   a. In Sync message: Requires on-the-fly message modification
   b. In a Follow_Up message: Easy to insert but requires IEEE 1588 code to keep track of pairs of messages.

Accuracy Issues (hardware assist)

1. Hardware assisted generation of time stamps is potentially the most accurate.
2. Requires attention to latency (clause 6.2.4.9) and message time stamp point (clause 6.2.2.3)
3. In addition to capturing the time stamp enough information must be captured to enable IEEE 1588 code to associate the time stamp with the correct Sync message
4. Must differentiate between IEEE 1588 Sync (or Delay_Req) messages and other traffic.

Accuracy Issues (hardware assist)



Accuracy Issues (oscillators)

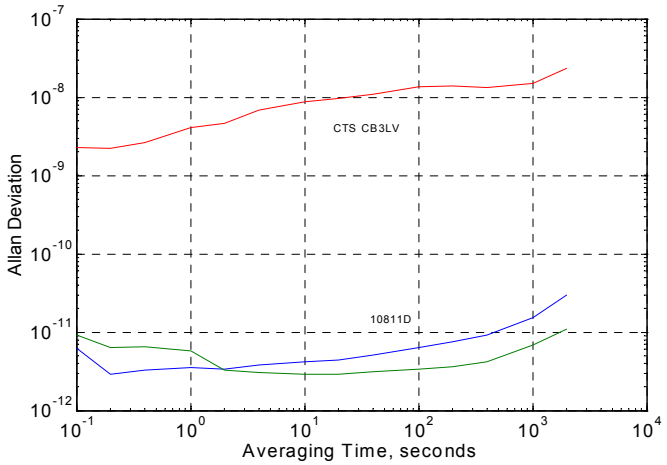
1. **IEEE 1588 is all about reducing timing fluctuations:**
   a. **In the protocol stacks: hardware assist**
   b. **In network components: boundary clocks**
2. **The final reduction technique is statistics:**
   a. **Pre-filtering of raw clock offset data**
   b. **Design of the servo in the slaves**
3. **Clocks must be sufficiently stable to support the statistic given sync_interval, fluctuation level, and desired accuracy.**

Accuracy Issues (oscillators)

Experience has shown:

- **Accuracy ~100 ns level is achievable with 2 second updates, inexpensive oscillators, compact topologies with lightly loaded switches, and simple PI servos for averaging.**
- **Accuracy <20 ns will require some combination of faster sampling, better oscillators, boundary clocks, sophisticated statistics and servo algorithms and careful control of environment especially temperature.**

Allan Frequency Deviations for Two Oscillators



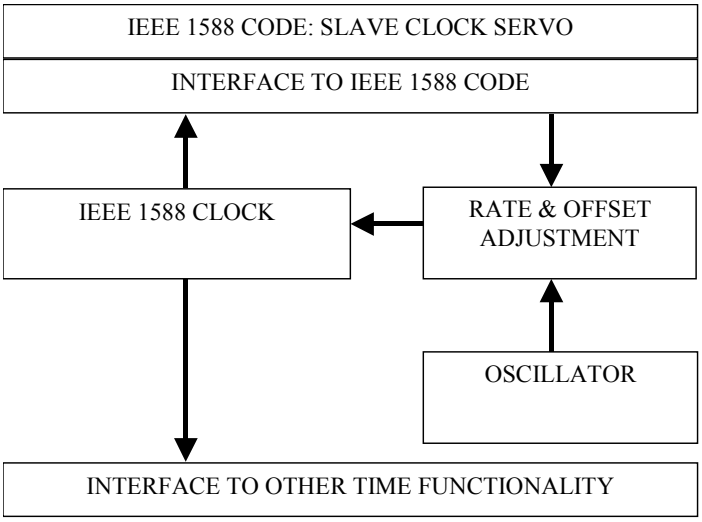
Accuracy Issues (asymmetry)

1. Path asymmetry introduces offset errors
2. Whether asymmetry needs to be considered depends on the network topology and implementation and the desired accuracy
3. The major source of asymmetry in a complex network is different path lengths in the master to slave and slave to master directions. This can result from queuing differences in switches/routers or in actual routing differences.
   a. Control routing
   b. Measure and correct for delay

Accuracy Issues (asymmetry)

4. Physical media can also be asymmetric
   a. CAT5 cable asymmetry is nominally 25-50ns/100m
   b. Measure and correct for delay

Accuracy Issues (clock design)



Clock Design Issues

1. **IEEE 1588 Clock**
   a. **Width: how many bits of seconds, resolution, rollover (Y2K)**
   b. **Representation: binary, BCD, sec/ns vs. ns**
2. **Rate & offset adjustment**
   a. **Rate range: must allow for maximum offset specification on oscillators (+-0.01%)**
   b. **Minimum correction: Consistent with desired accuracy, e.g. 1 part in 10^9**
   c. **Offset correction: Must allow gross error correction on transients**
3. **Interface to IEEE 1588 code**
4. **Interface to other time functionality**

Clock Design Issues

IEEE 1588 Code: Slave clock servo

1. **Servo input is the 'offset' computed from time stamps of Sync and Delay_Req messages exchanged between master and slave**
2. **Typical implementations of the servo use a PI (proportional integral) control strategy**
3. **Usual issues of servo stability, parameters, wind-up, outliers in error input,…**

Clock Design Issues

IEEE 1588 Code: Grandmaster clock

1. **The grandmaster clock determines the time base for the entire system.**
2. **The grandmaster clock MAY itself synchronize to a source of time EXTERNAL TO THE IEEE 1588 system:**
   a. **Application time base (within the tolerance of the IEEE 1588 system)**
   b. **GPS, NTP, or other recognized UTC time base. In all cases but especially with NTP a 'flywheel' will be needed to average out fluctuations of the source to the desired accuracy of the IEEE 1588 system.**

Clock Design Issues

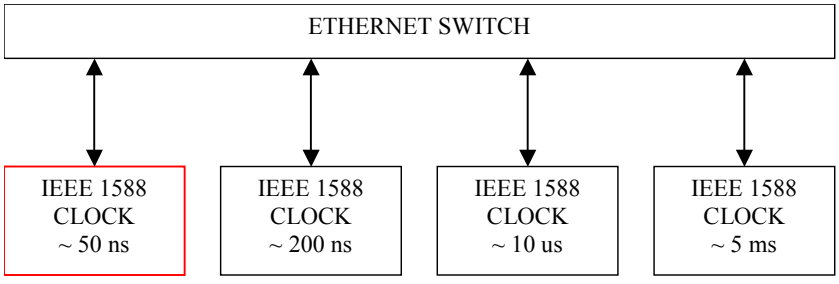
IEEE 1588 Code: Grandmaster clock (continued)

Synchronization to an external source can be implemented using the clock servo normally used when in the slave state by:

1. **Generating an error signal representing the offset between the IEEE 1588 grandmaster clock and the external source. External sources, e.g. GPS, typically provide a 1 PPS signal useful for this purpose.**
2. **Applying the error signal to the grandmaster clock servo.**

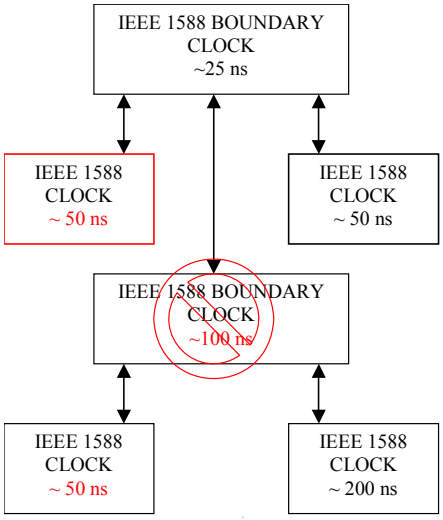
Accuracy Issues (topology)

Single subnet: no problem



Accuracy Issues (topology)

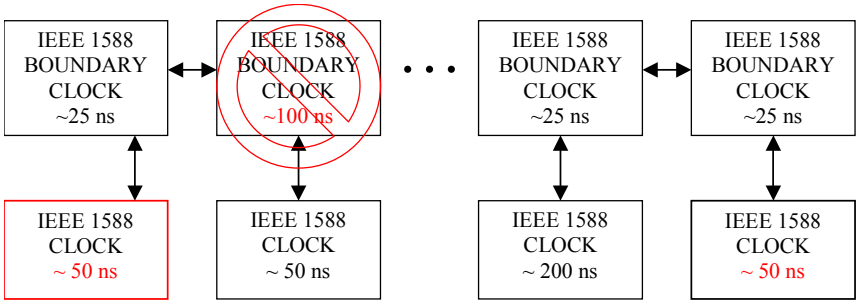
Hierarchy: Be Careful!



Accuracy Issues (topology)

Linear: Be Careful!

1. **Cascaded devices accumulate servo error & quantization errors**
2. **Low accuracy intermediate devices dominate error budget of chain**



Application Level Support

How do applications interface to an IEEE 1588 clock?

1.Time stamp events

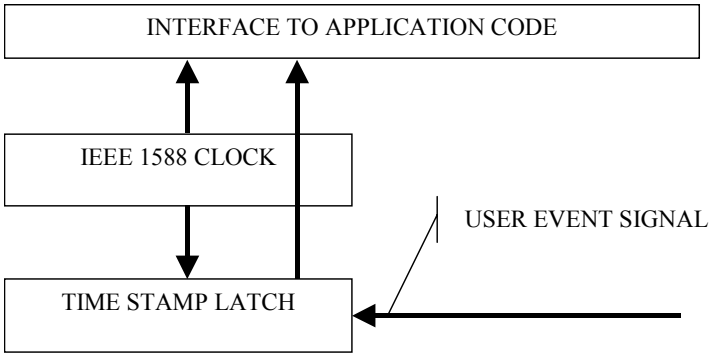
2.Generate events

3.Generate waveforms

In each case the application signals in one device will be correlated in time with those in other devices within the synchronization accuracy of the underlying IEEE 1588 clocks.

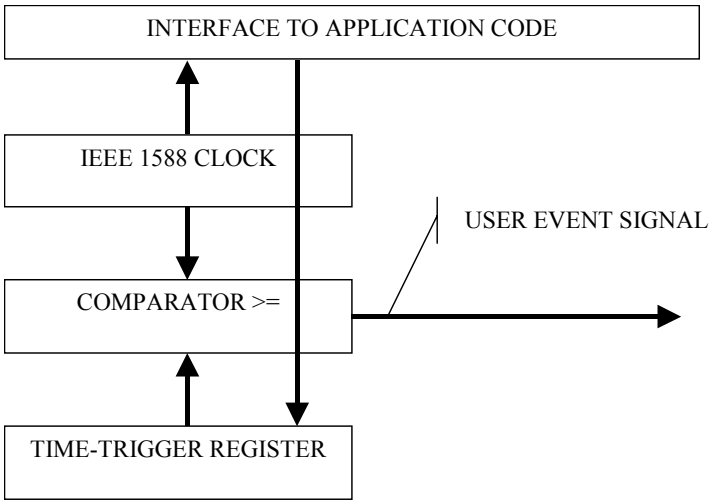
Application Level Support

How to time stamp events:



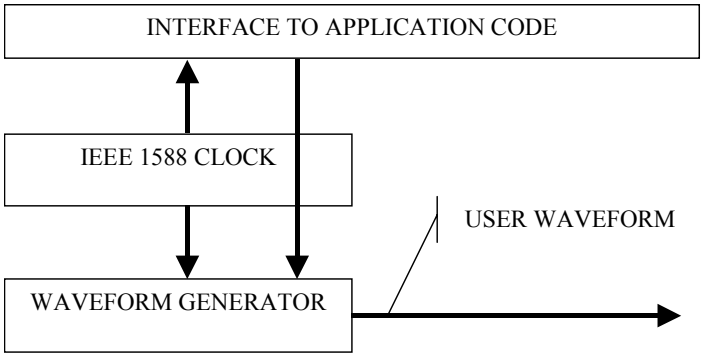
Application Level Support

How to generate events (time-triggers):



Application Level Support

How to generate waveforms:



Extensions to IEEE 1588 version 1 in PAR of the P1588 Committee

P1588 PAR Topics

1. **Resolution of known errors: A list of these and recommended solutions is posted on the IEEE 1588 web site. http://ieee1588.nist.gov These are not expected to have appreciable impact on existing implementations.**
2. **Conformance enhancements: 1 PPS or equivalent signal, management message or extension fields to make internal time stamps visible.**
3. **Enhancements for increased resolution and accuracy:**
   - **Extension fields to allow sub-nanosecond time stamps,**
   - **shorter sync_intervals allowed.**
4. **Increased system management capability: Additional management messages, perhaps SNMP**

(items in red may substantially impact version 1 operation or compatibility)

P1588 PAR Topics (continued)

5. Mapping to DeviceNet: Few if any changes required in body of standard
6. Annex D modifications for variable Ethernet headers: Likely additions are tagged frames and IPV6. These could impact existing packet recognition designs and protocol stacks.
7. Prevention of error accumulation in cascaded topologies: New clock type (transparent clock), topology and system design guidelines.
8. Rapid network reconfiguration: Path delay measurements and correction of timestamps.
9. Ethernet layer 2 mapping

P1588 PAR Topics (continued)

10. Optional shorter frame: Must resolve needs of industrial and telecommunication applications.
11. Extensions to enable implementation of redundant systems:
    - Master clock failure and network failure.
    - Redundant grandmaster clocks, and/or
    - Slave selection of grandmaster clocks.
12. Security extensions: authentication of grandmaster,…
13. Extension mechanism: Uniform way of extending fields/messages.

IEEE Procedures to Revise/Update the Standard

1. **IEEE sponsor (Kang Lee for TC-9 of I&M Society) appoints chair of working group.**
2. **Solicit membership in working group.**
3. **Draft and submit PAR (project authorization request) to the IEEE**
4. **PAR approval (March, 2005)**
5. **Develop revised standard (12-18 months)**
6. **Submit to IEEE ballot process (~ 3 months)**
7. **Revise/re-ballot if necessary**
8. **Editorial/publish process with IEEE (~ 3 months)**

Application Areas

a. **Industrial automation**

b. **Telecommunications**

c. **Test and measurement**

Industrial Automation
Anatoly Moldovansky- Rockwell
Ludwig Winkel- Siemens

Telecommunications
Silvana Rodrigues- Zarlink

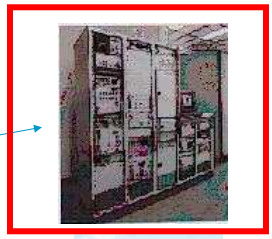
Test and Measurement
John Eidson- Agilent

Test and Measurement

1. **Moving from bus (IEEE-488 aka. GPIB) connected instrument systems to network connected modular systems.**
2. **Synchronization needs vary widely with application**
   a. **Low to sub-nanosecond for most demanding**
   b. **Microseconds to milliseconds for less demanding**

Military Systems

1. Variety of potential applications
   a. Depot and test ranges
   b. Flight test & qualification
   c. Operational systems
2. Requirements very similar to test and measurement



LXI Consortium

•Consortium of test and measurement equipment vendors and users

•LXI Specification:

- Mandates the use of IEEE 1588 for LXI Class B instrumentation
- Specifications on how to use IEEE 1588 in instruments
  - Timestamp data and events
  - Time-triggers
  - Peer-peer LAN messages containing event timestamps

- LXI paper during this conference.

Styles of Measurement and Control

a. **Message based**

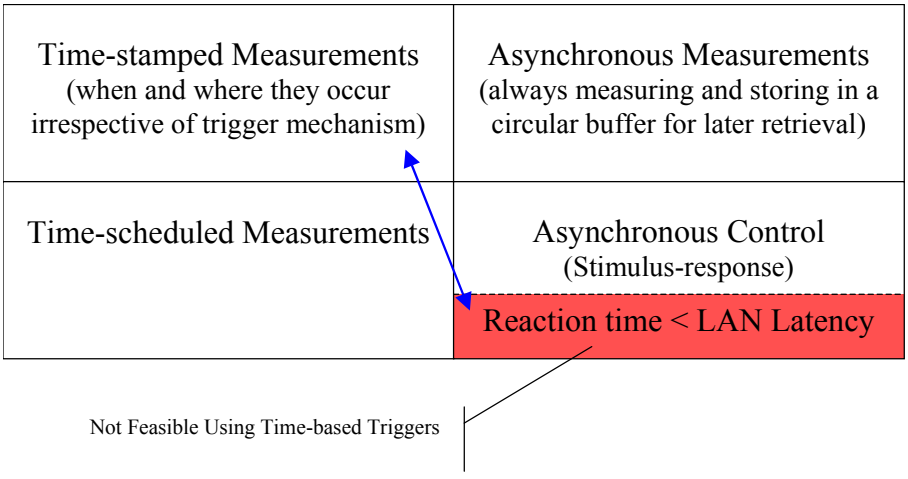
b. **Periodic**

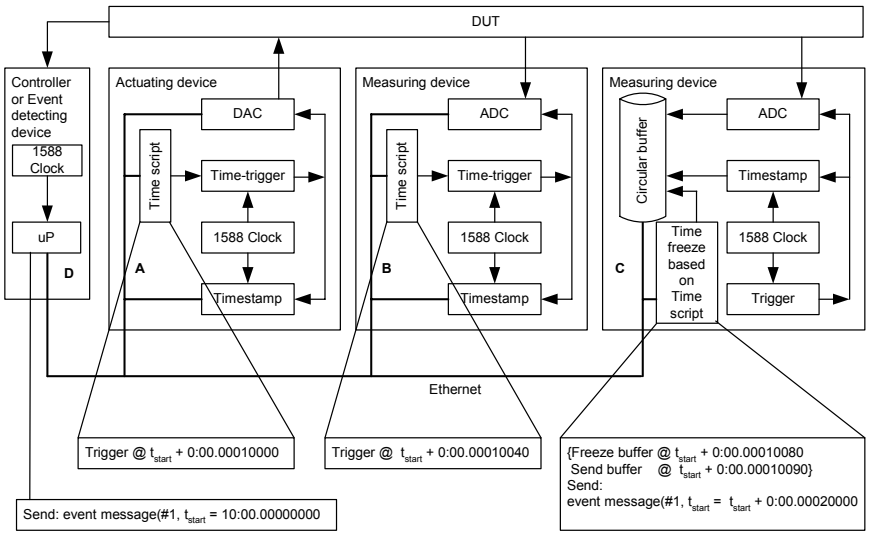
c. **Time-based**

Styles of Control

	Message-based	Cyclic	Time-based
Information dependent on message	Value and timing	Value and timing	Value and time specification
Timing accuracy limited by:	Fluctuations in message generation timing and delivery latency	Fluctuations in cycle periodicity	Accuracy of clock synchronization
Update timing resolution limited by	Latency and minimum inter-message interval	Cycle period	Resolution of the clock
Ordering of data to/from multiple sources	Dependent on messaging protocol	Tied to cycle	Limited by synchronization accuracy and clock resolution

Test and Measurement Application Space




DUT
Controller or Event detecting device
1588 Clock
uP
D
Actuating device
DAC
Time script
Time-trigger
1588 Clock
Timestamp
A
Measuring device
ADC
Time script
Time-trigger
1588 Clock
Timestamp
B
Measuring device
Circular buffer
ADC
Timestamp
Time freeze based on Time script
1588 Clock
Trigger
C
Ethernet
Trigger @ t_start + 0:00.00010000
Trigger @ t_start + 0:00.00010040
{Freeze buffer @ t_start + 0:00.00010080
Send buffer @ t_start + 0:00.00010090}
Send:
event message(#1, t_start = t_start + 0:00.00020000
Send: event message(#1, t_start = 10:00.00000000

Questions?

IEEE-1588™ Telecommunications Applications

Silvana Rodrigues
Phone: +1 613 2707258
silvana.rodrigues@zarlink.com
http://timing.zarlink.com/

Antti Pietilainen
Phone: +358(0)718036660
antti.pietilainen@nokia.com
http://www.nokia.com

AGENDA

- **Telecommunication Synchronization Background**
  - **Telecom Synchronization**
  - **North America and International Telecommunication Union (ITU-T) Timing Distribution Hierarchy**
  - **Synchronous and Converged network model**
- **Telecom Applications Examples using 1588**
- **IEEE-1588™ Standard work to address Telecom Applications**
  - **IEEE-1588 Issues for Telecom**
  - **IEEE-1588 Enhancements to support Telecom**
  - **IEEE-1588 Standard work to support Telecom**
- **Summary**

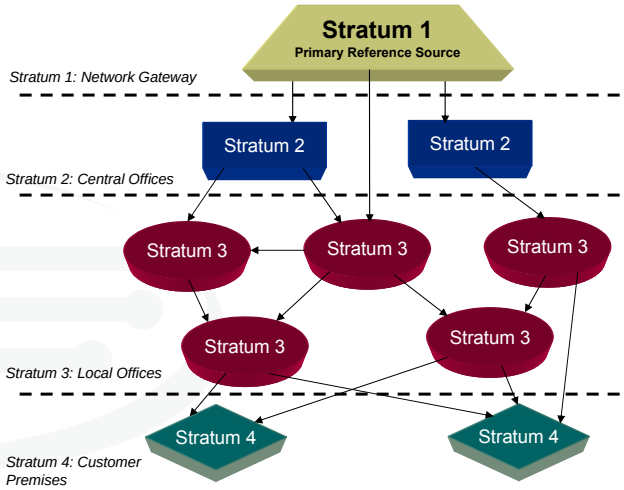
IEEE-1588™ is trademark of its respective owner

Telecommunication Synchronization Background

Telecom Synchronization

- **Clock quality levels (stratum for North America and Types and Options for the International Telecommunication Union - ITU) are defined by the industry standards organizations to maintain clock quality in the network**
- **Time sensitive services need synchronization**
- **Synchronization is important to avoid overflow or underflow of slip buffers, bit errors and other adverse effects**
  - **ITU-T Recommendation G.822 provides criteria for controlled slip rate**

North America Timing Distribution Hierarchy

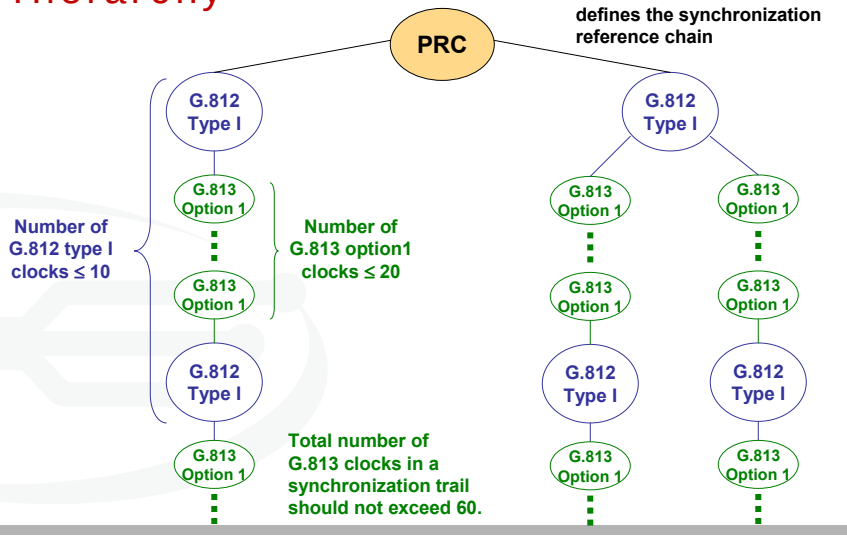


[Page 4]

ZARLINK SEMICONDUCTOR

ITU-T SDH Timing Distribution Hierarchy

ITU-T Recommendation G.803 defines the synchronization reference chain



[Page 5]

ZARLINK SEMICONDUCTOR

Clock Level

North America Stratum Level	ITU-T Clock Level	Free-run Accuracy	Holdover Stability	Pull-in/ Hold-in range	Wander Filtering	Phase Transient (Re-arrangement)
1 (PRS)	PRC (G.811)	+/- $1x10^{-11}$	N/A	N/A	N/A	N/A
2	Type II (G.812)	+/- 0.016 ppm	+/- $1x10^{-10}$ /day	0.016 ppm	0.001Hz	MTIE < 150ns
Not Defined	Type I (G.812)	N/D	+/- $2.7x10^{-9}$ /day	0.01 ppm	0.003Hz	MTIE < 1µs
3E	Type III (G.812)	+/- 4.6 ppm	+/- $1.2x10^{-8}$ /day	4.6 ppm	0.001Hz	MTIE < 150ns Phase slope 885ns/s
3	Type IV (G.812)	+/- 4.6 ppm	+/- $3.9x10^{-7}$ /day	4.6 ppm	3Hz 0.1Hz (SONET)	MTIE < 1µs Phase slope 61us/s Objective: MTIE < 150n Phase slope 885ns/s
Not Defined	Option I (G.813)	+/- 4.6 ppm	+/- $2x10^{-6}$ /day	4.6 ppm	1 – 10Hz	MTIE < 1µs
SMC	Option 2 (G.813)	+/- 20 ppm	+/- $4.6x10^{-6}$ /day	20 ppm	0.1Hz	MTIE < 1µs Objective mask 150ns Phase slope 885ns/s
4	4	+/- 32 ppm	N/A	32 ppm	No	No Requirement

[Page 6]

ZARLINK SEMICONDUCTOR

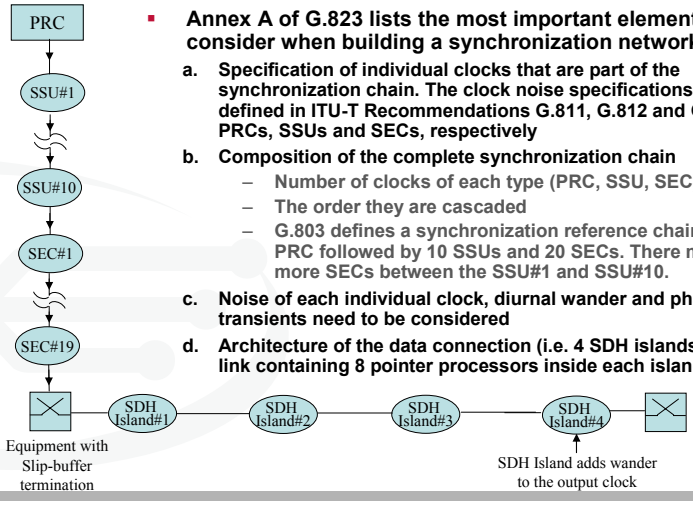
Standard Requirements

- **ITU-T recommendations, G.823 for E circuits and G.824 for T circuits set limits on the magnitude of jitter and wander at network interfaces. The wander may not exceed given values anywhere in the network. Thus, a circuit emulation link, for example, may consume only part of the wander budget**
- **GSM, WCDMA, and CDMA2000 require 0.05 ppm at air interface**
- **CDMA2000 requires time synchronization at ± 3 µs level (±10 µs worst case)**
- **WCDMA TDD mode requires 2.5-µs time accuracy between neighboring base stations (i.e. ±1.25 µs of UTC)**
  - **These requirements are too difficult to achieve without good transparent clocks or boundary clocks in each intermediate node**
  - **Some cellular operators do have control over the transport network so they could use IEEE1588 compliant switches for achieving time synchronization**

[Page 7]

ZARLINK SEMICONDUCTOR

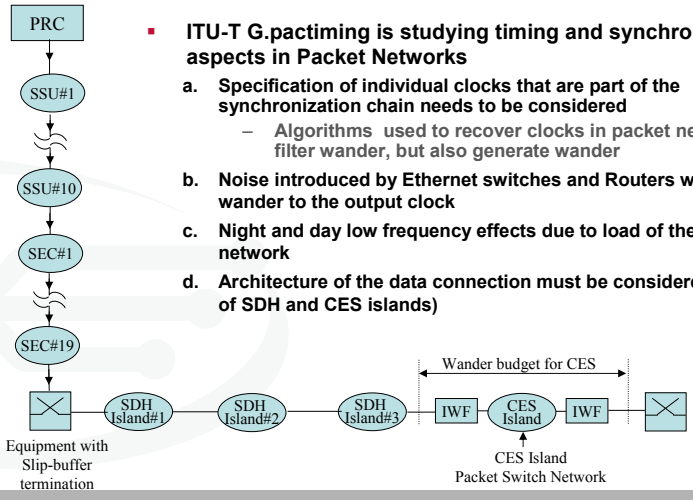
Synchronous Network Model



- **Annex A of G.823 lists the most important elements to consider when building a synchronization network**
  - **a. Specification of individual clocks that are part of the synchronization chain. The clock noise specifications are defined in ITU-T Recommendations G.811, G.812 and G.813 for PRCs, SSUs and SECs, respectively**
  - **b. Composition of the complete synchronization chain**
    - Number of clocks of each type (PRC, SSU, SEC)
    - The order they are cascaded
    - G.803 defines a synchronization reference chain with 1 PRC followed by 10 SSUs and 20 SECs. There may be 40 more SECs between the SSU#1 and SSU#10.
  - **c. Noise of each individual clock, diurnal wander and phase transients need to be considered**
  - **d. Architecture of the data connection (i.e. 4 SDH islands on the link containing 8 pointer processors inside each island)**

ZARLINK SEMICONDUCTOR

Converged Network Model



- **ITU-T G.pactiming is studying timing and synchronization aspects in Packet Networks**
  - **a. Specification of individual clocks that are part of the synchronization chain needs to be considered**
    - Algorithms used to recover clocks in packet networks filter wander, but also generate wander
  - **b. Noise introduced by Ethernet switches and Routers will add wander to the output clock**
  - **c. Night and day low frequency effects due to load of the packet network**
  - **d. Architecture of the data connection must be considered (i.e. mix of SDH and CES islands)**

ZARLINK SEMICONDUCTOR

Telecom Applications Examples using 1588

ZARLINK SEMICONDUCTOR

Requirement scenarios

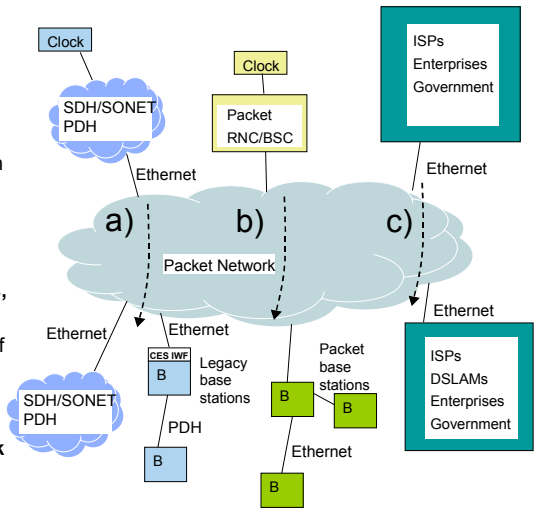
a) **Connecting SDH/SONET/PDH nodes and networks (circuit emulation).**

The connections between SDH/SONET/PDH nodes may be leased from another carrier (e.g. cellular operators usually do not own the transport network). Typical requirements are to meet ITU-T G.823 and G.824.

b) **Connecting nodes, which require synchronization for other reasons, e.g. cellular base stations.**

Typical requirements are 0.05ppm of frequency accuracy.

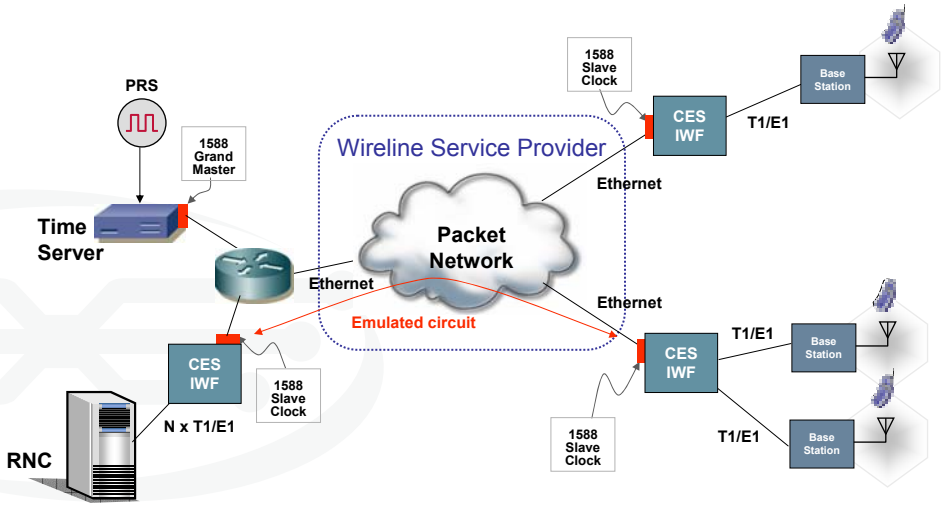
c) **Connecting offices and nodes of Internet service providers (ISPs), enterprises, government. The bulk of all traffic.**



[Page 11]

ZARLINK SEMICONDUCTOR

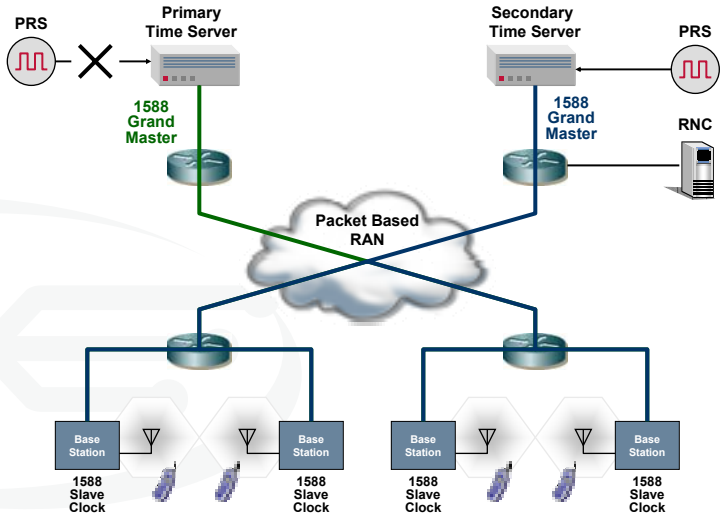
IEEE-1588 used in CES Application



[Page 12]

ZARLINK SEMICONDUCTOR

IEEE-1588* used in Wireless Networks



* With proper changes to the current standard

[Page 13]

ZARLINK SEMICONDUCTOR

1588 Standard Work to Support Telecom

ZARLINK SEMICONDUCTOR

IEEE-1588 Issues for Telecom

- **IEEE-1588 only allows the values of sync interval to be 1, 2, 8, 16, and 64 seconds**
  - **It is difficult to maintain performance in a loaded network with sync packet rate of 1pps and an inexpensive oscillator**
- **IEEE-1588 relies on a symmetric network**
- **IEEE-1588 does not have provision for redundancy support**
  - **In telecom applications clocks must be always available**
- **IEEE-1588 relies on boundary clocks topology**
  - **Boundary clocks are not available in legacy telecom networks**
- **IEEE-1588 only supports multicast**
- **IEEE-1588 Message Format**
  - **Long PTP messages consuming too much bandwidth**

[Page 15]

ZARLINK SEMICONDUCTOR

IEEE-1588 Enhancements to Support Telecom

- **Enhancements for increased resolution and accuracy**
  - **Allow shorter sync_intervals**
- **Extensions to the standard to enable correction for asymmetry**
- **Extensions to the standard to enable implementation of redundant systems – Fault Tolerant Systems**
  - **Deal with master clock failure and network failure**
- **Prevention of errors accumulation in cascaded topologies**
  - **Deal with boundary clock issues for telecom applications**
- **Use of Unicast in addition to Multicast**
- **Short Frame, reduced message format**
- **Support for QoS**

ZARLINK SEMICONDUCTOR

IEEE-1588 Standard Work to Support Telecom

- **Short Frame Format**
  - **There is a consensus to have four short frame messages**
    - **Short Sync Message**
    - **Short Follow-up Message**
    - **Short Delay_Req Message**
    - **Short Delay_Resp Message**
  - **The short frame protocol allows shorter sync_intervals**
  - **The short frame protocol supports a mixed of short and long messages**
  - **The current long frame format is still used for the Best Master Clock algorithm and also to allow slaves to find the address and status of available masters**
  - **The existing Delay Request and Delay Response messages no longer need to be transmitted**
  - **The short messages give the same timing information as the long messages of the existing standard and use the same timestamp format**
  - **The short frame protocol allows the slave to vary the rate at which it receives time information according to its needs**

ZARLINK SEMICONDUCTOR

IEEE-1588 Standard Work to Support Telecom cont'd

- **Fault Tolerant**
  - **There are 3 proposals**
    - Two slave centric proposals and one master centric proposal
  - **Fault Tolerant Goals**
    - The fault of any single network element can not cause slaves to experience a sudden phase change.
    - A faulty grand master should be detected and replaced rapidly by another grand master.
    - Switching from one grand master to another should not result in a significant phase step at the slaves
  - **Fault Tolerant subcommittee is working on a single proposal that aligns all the 3 proposals**

ZARLINK SEMICONDUCTOR

Summary

- **The interest on IEEE1588 in the Telecom Industry is growing**
- **Several applications within Telecom can benefit from a Precision Clock Synchronization Protocol like IEEE1588**
- **The work in IEEE1588 to support Telecom is progressing**
  - **Short Frame Format is stable**
  - **Fault Tolerant work is on going**
  - **Still several issues that need work**
    - Issues must be resolved in a timely matter
    - It should be avoided (as much as possible) to add complex functionality to the standard

ZARLINK SEMICONDUCTOR

Acronyms

- **PRC** Primary Reference Clock
- **PRS** Primary Reference Source
- **SDH** Synchronous Digital Hierarchy
- **SEC** SDH Equipment Clock
- **SSU** Synchronization Supply Unit
- **PDH** Plesiochronous Digital Hierarchy
- **GSM** Global System for Communications
- **CDMA** Code Division Multiple Access
- **WCDMA** Wide-band CDMA
- **TDD** Time Division Duplex
- **RNC** Radio Network Controller (WCDMA)
- **BSC** Base Station Controller (GSM)
- **DSLAM** Digital Subscriber Line Access Multiplexer

ZARLINK SEMICONDUCTOR

Thank you!

ZARLINK SEMICONDUCTOR

Application of IEEE 1588 in Industrial Automation and Motion Control Systems

Anatoly Moldovansky
Rockwell Automation
October 10, 2005

Using time for control…

…not just network-based events!

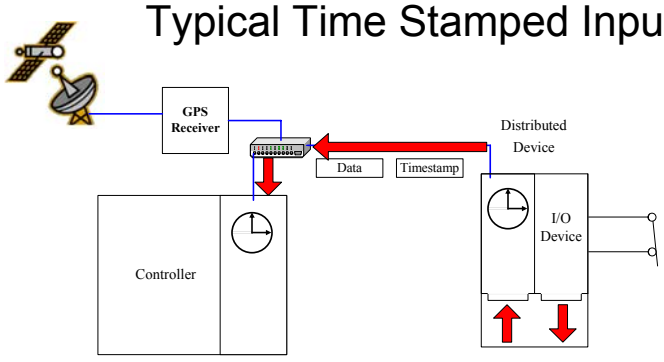
IEEE 1588 Provides Time Synchronization Services

- Synchronization Services
  - The industrial market is driving the need for synchronization to a common time-base with sub-microsecond accuracy, node-to-node.
- IEEE 1588
  - Nanosecond Clock Resolution
  - +/- 100 nanosecond, or better, clock synchronization between distributed devices

Applications for Time Synchronization

- **Sequence of Events Measurements**
- **Scheduled Outputs**
- **Synchronized Actuation**
- **Time-Stamped Data Logging**
- **Coordination with GPS Time**

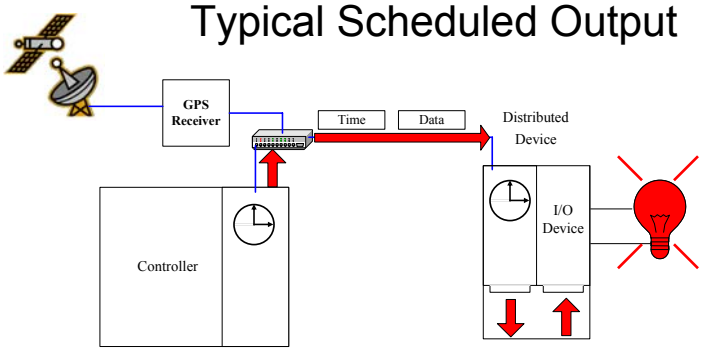
Typical Time Stamped Input



Applications for Time Synchronization

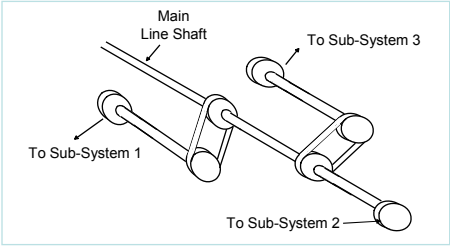
- **Sequence of Events Measurements**
- **Scheduled Outputs**
- **Synchronized Actuation**
- **Time-Stamped Data Logging**
- **Coordination with GPS Time**

Typical Scheduled Output



Distributed Motion Control

- Today's distributed motion control applications are founded in mechanical line shafting designs. A single mechanical line shaft drives multiple subsystems using belts, pulleys or gear boxes.



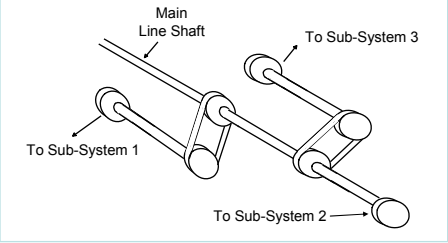
- Typically, these applications are characterized as phase locked - or "lineshaft" applications. Like a large music box, all mechanical elements are timed and phased through mechanical means.

Distributed Motion Control

- Mechanical Lineshafts are inflexible
  - Single product design
  - Long product change-over
  - Run-time adjustments for re-phasing were non-existent or required expensive differential gear-boxes.
  - Wear and tear of mechanical components
- Much power was expended on moving machinery and not product.

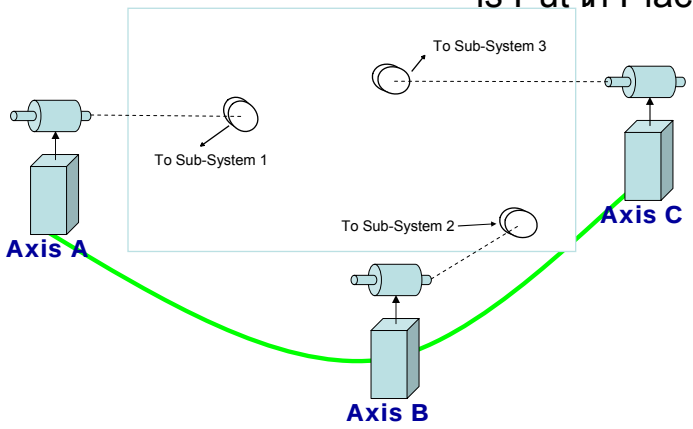
Distributed Motion Control

- Mechanical designs have given way to electronic design control schemes



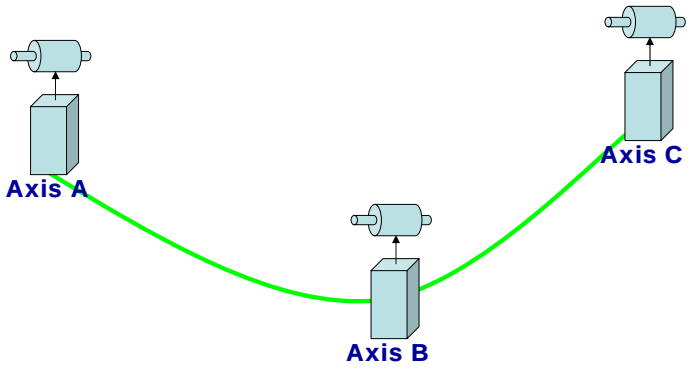
Distributed Motion Control

A Communications Network is Put in Place…



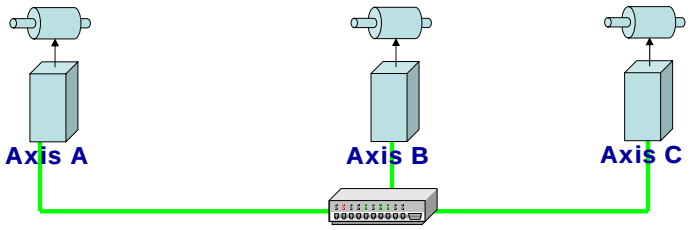
Distributed Motion Control

And the Result is an Electronic LineShaft!



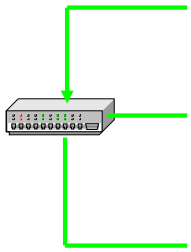
Distributed Motion Control

- And the Result is an Electronic Lineshaft!



Why is Time Synchronization Required?

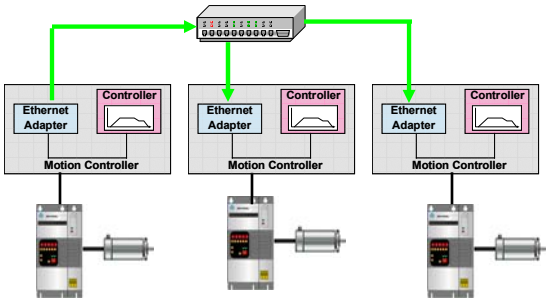
- Each Motion Controller Controls *Position over Time*



CIP Motion™

- There are Two Types of Connections that are Typically Used for Distributed Motion Control

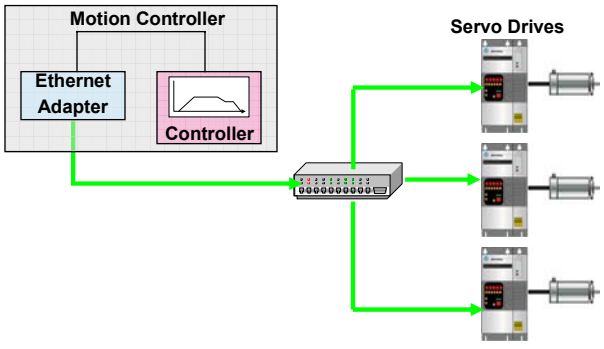
Peer to Peer



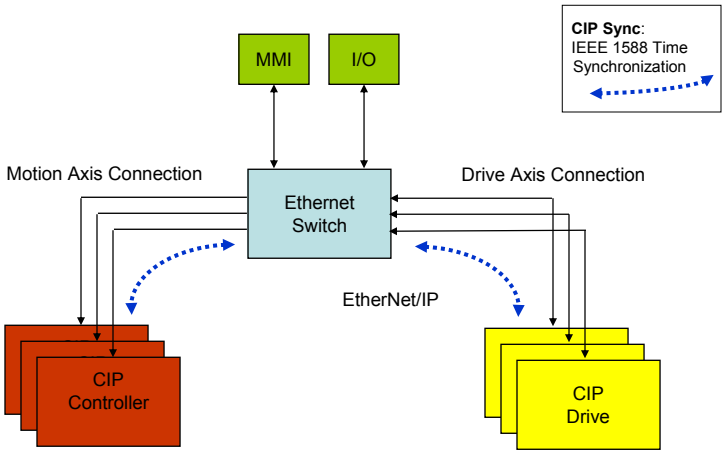
CIP Motion™

- There are Two Types of Connections that are Typically Used for Distributed Motion Control

Control to Drive

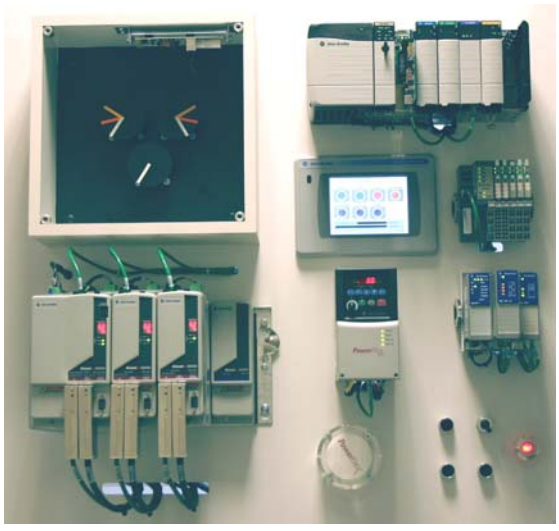


CIP Motion Architecture



Demo

CIP Motion™ Demo



IEEE 1588 Workshop Tutorial

Industrial and Motion Control Applications

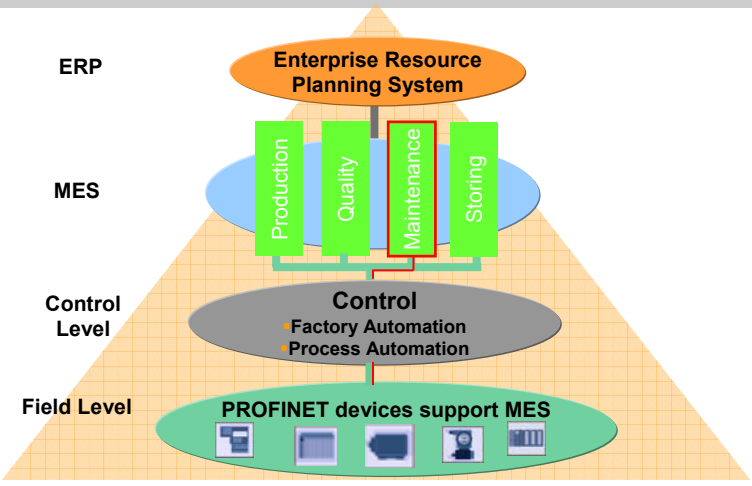
- Introduction to industrial automation
- Communication networks
- Applications
- PROFINET

SIEMENS

Automation hierarchy

IEEE 1588
Industrial and Motion Control Applications

Introduction
Network
Applications
PROFINET

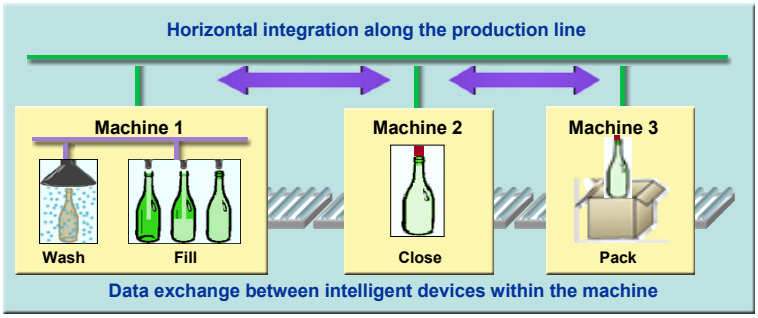


SIEMENS

© Siemens AG 2005

Distributed Automation – Plant View Modular Plant and Machine Construction

- **Example from the food & beverage industry:**
  - **Wash bottles**
  - **Fill bottles**
  - **Close bottles**
  - **Pack bottles**



© Siemens AG 2005

Communication network Standards

IEC/PAS 62411 and IEC 61784-2

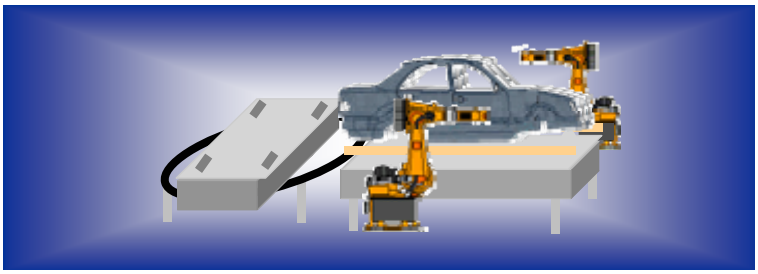
© Siemens AG 2005

Real-Time Ethernet (RTE) with PROFINET

IEEE 1588
Industrial and Motion Control Applications

Introduction
Network
Applications
PROFINET

PROFINET



Real-Time Communication

SIEMENS

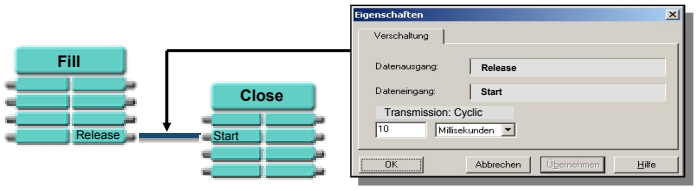
© Siemens AG 2005

PROFINET CBA: Real-Time between Components

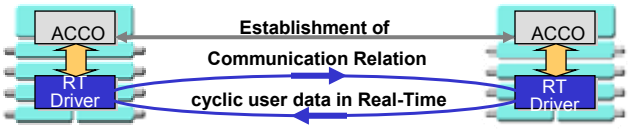
IEEE 1588
Industrial and Motion Control Applications

Introduction
Network
Applications
PROFINET

- **The user chooses the QoS „Real-Time Data Transmission“ in the configuration tool**



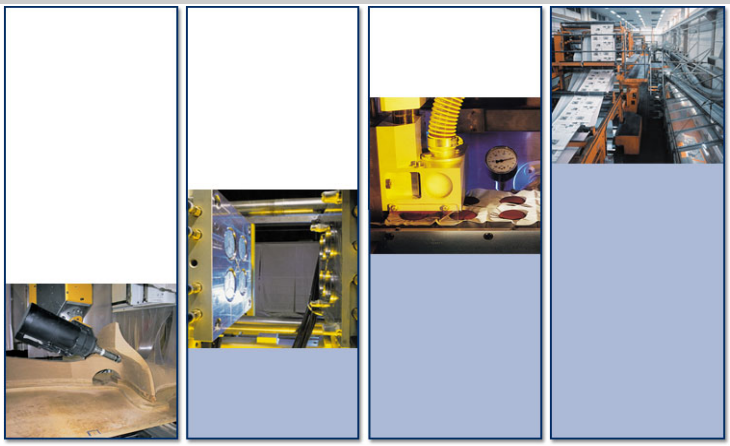
- **The Communication relationships between the devices is established over TCP/IP**
- **Subsequently, process data are transmitted cyclically between devices via the Real-Time channel**



SIEMENS

© Siemens AG 2005

Demands on Motion Control applications



Wood-, glass- and ceramic-processing machines | Plastics injection molding machines | Packaging machines | Printing presses

© Siemens AG 2005

Trends

- **Time stamping from sensor level to HMI**
- **Precise clock synchronization**
  - **With bridges (e.g. IE → PB) actually 10 ms plant wide**
  - **Industrial Ethernet (IE) actually 1 ms plant wide**
  - **Both are to enhance**
- **Enhanced diagnosis required with precise time stamping**
- **A plant wide reliable synchronization source**
- **Robots synchronized using clock synchronization**
- **Clock synchronization protocols:**
  - **NTP in cell level (HMI, EMS, ERP)**
  - **PTP (IEEE 1588) in field level (actor/sensor + control)**

© Siemens AG 2005

Application Requirements

- **Chronological association of diagnosis and process alarms**
- **Time dependent process synchronization**
  - **Net diagnosis on switch port with time stamp**
- **log files with time stamp**
  - **Security log files (IP-ACL)**
  - **Configuration log files**
  - **Device log files**
- **Clock synchronization precision plant wide below 1ms**
  - **IP-sub-net included**
- **Standby-Clock master**
- **Alerts for clock master failures**
- **Summer/Winter-time adjust independent of clock synchronization protocol**

© Siemens AG 2005

Real-Time Communication Classes

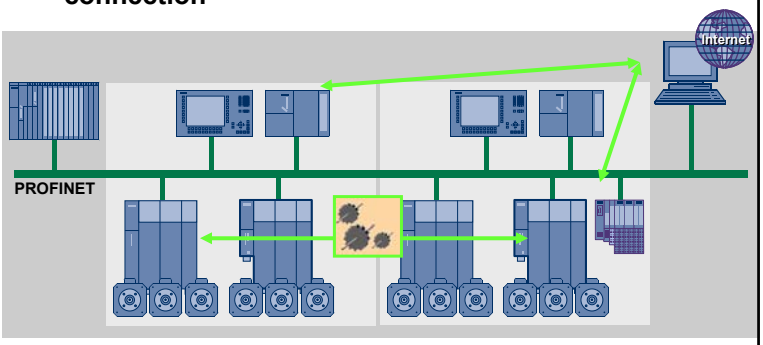
- **PROFINET distinguishes between two real-time classes with differences regarding the performance:**
- **Real-Time:**
  - **Using standard components**
  - **Performance characteristics like fieldlbuses today (e.g. PROFIBUS)**
  - **Typical application area: Factory Automation**
- **Isochronous Real-Time:**
  - **Clock synchronized communication**
  - **Hardware support via Switch-ASIC**
  - **Typical application area: drive control in Motion Control applications**

© Siemens AG 2005

Motion Control with PROFINET

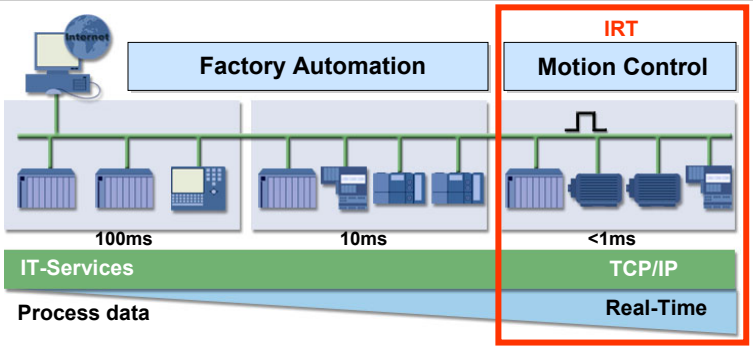
Advantages at a glance

- **Isochronous communication for Motion Control Applications**
- **Short and deterministic reaction times of < 1ms, Jitter < 1µs**
- **Integration of decentralized field devices**
- **TCP/IP for engineering, diagnostics and HMI connection**



© Siemens AG 2005

Isochronous Real-Time Communication (IRT)



- **Requirements on Ethernet for Motion Control**
  - **Highest performance**
  - **Time synchronization inclusive determinism**
  - **Openness for unrestricted access to the IT world, which means no restrictions for TCP/IP**

© Siemens AG 2005

PROFINET and IRT

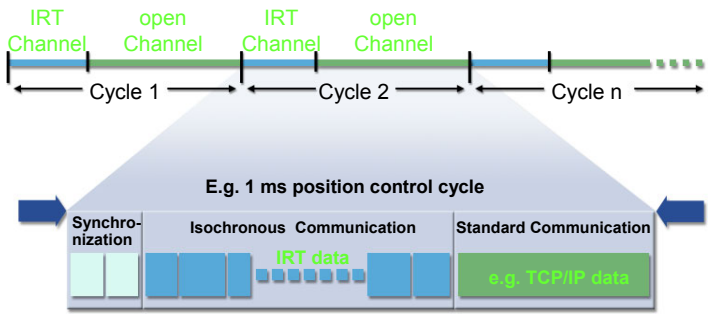
What are the pre-conditions ?

- **Segmentation of the communication**
- **use of time based communication**
- **Clock-Synchronization**

© Siemens AG 2005

IRT Scheduling

- **Scheduling of communication systems**
  - **High accurate cycle synchronization**
  - **Separate time areas for real-time and TCP/UDP**



© Siemens AG 2005

IEEE-1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

-Test and Measurement Applications-

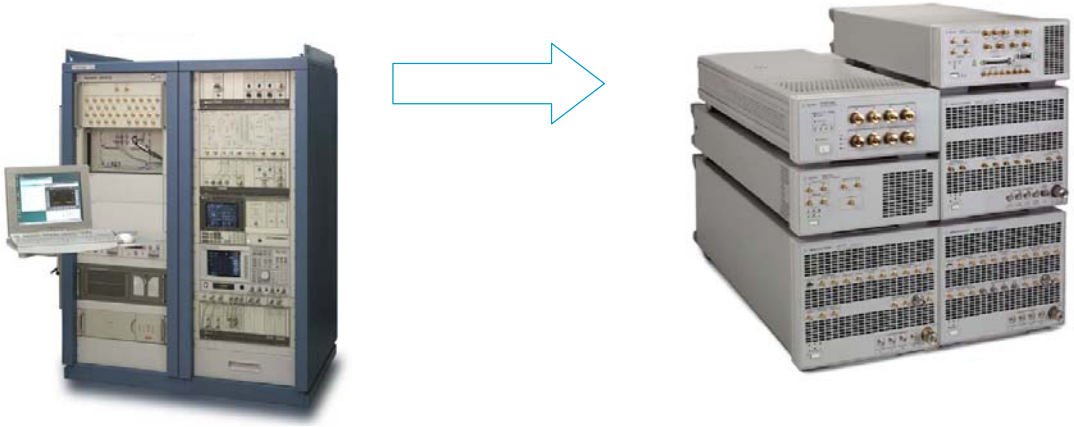
John Eidson
October 10, 2005
john_eidson@agilent.com

Agilent Technologies

© Copyright 2005 Agilent Technologies, Inc

Test and Measurement

1. Moving from bus (IEEE-488 aka. GPIB) connected instrument systems to network connected modular systems.
2. Synchronization needs vary widely with application
   a. Low to sub-nanosecond for most demanding
   b. Microseconds to milliseconds for less demanding



Military Systems

1. Variety of potential applications
   a. Depot and test ranges
   b. Flight test & qualification
   c. Operational systems
2. Requirements very similar to test and measurement

Agilent Technologies

LXI Consortium

•Consortium of test and measurement equipment vendors and users

•LXI Specification:

- Mandates the use of IEEE 1588 for LXI Class B instrumentation
- Specifications on how to use IEEE 1588 in instruments
  - Timestamp data and events
  - Time-triggers
  - Peer-peer LAN messages containing event timestamps

- LXI paper during this conference.

Agilent Technologies

Styles of Measurement and Control

a. **Message based**

b. **Periodic**

c. **Time-based**

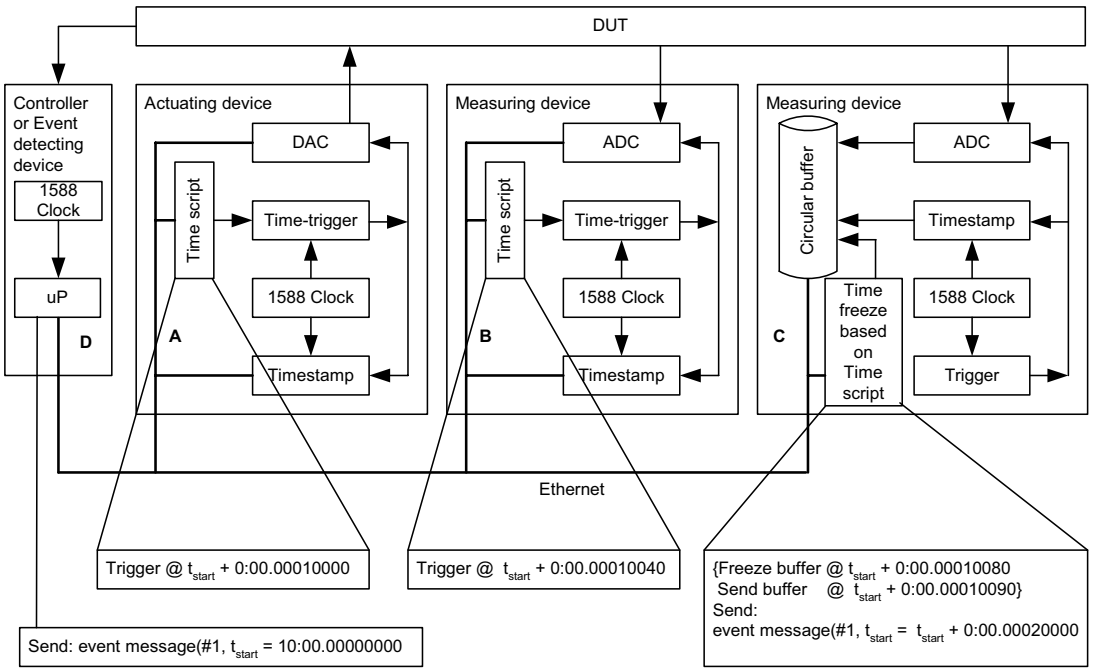
Styles of Control

	Message-based	Cyclic	Time-based
Information dependent on message	Value and timing	Value and timing	Value and time specification
Timing accuracy limited by:	Fluctuations in message generation timing and delivery latency	Fluctuations in cycle periodicity	Accuracy of clock synchronization
Update timing resolution limited by	Latency and minimum inter-message interval	Cycle period	Resolution of the clock
Ordering of data to/from multiple sources	Dependent on messaging protocol	Tied to cycle	Limited by synchronization accuracy and clock resolution

Test and Measurement Application Space

Time-stamped Measurements (when and where they occur irrespective of trigger mechanism)	Asynchronous Measurements (always measuring and storing in a circular buffer for later retrieval)
Time-scheduled Measurements	Asynchronous Control (Stimulus-response) Reaction time < LAN Latency

Not Feasible Using Time-based Triggers


DUT
Controller or Event detecting device
1588 Clock
uP
D
Actuating device
DAC
Time script
Time-trigger
1588 Clock
Timestamp
A
Measuring device
ADC
Time script
Time-trigger
1588 Clock
Timestamp
B
Measuring device
Circular buffer
ADC
Timestamp
Time freeze based on Time script
1588 Clock
Trigger
C
Ethernet
Trigger @ t_start + 0:00.00010000
Trigger @ t_start + 0:00.00010040
{Freeze buffer @ t_start + 0:00.00010080
Send buffer @ t_start + 0:00.00010090}
Send:
event message(#1, t_start = t_start + 0:00.00020000
Send: event message(#1, t_start = 10:00.00000000

Questions?

Design Considerations for Software Only Implementations of the IEEE 1588 Precision Time Protocol

Kendall Correll, Nick Barendt
VXI Technology, Inc.
Cleveland, Ohio, USA

Michael Branicky
EECS Dept., Case Western Reserve University
Cleveland, Ohio, USA

Abstract – This paper investigates adjusting computer clock frequency and time to provide a precise clock for test and measurement systems. In particular, it is concerned with the precision achievable using IEEE 1588 Precision Time Protocol systems without the support of specialized hardware. This paper outlines the design of a free IEEE 1588 implementation named PTPd. Particular attention is paid to the design of the clock servo—the system that steers the clock rate. This paper evaluates the implementation by the precision of the time coordination between networked test and measurement systems.

I. INTRODUCTION

The IEEE 1588 Precision Time Protocol (PTP) [1] provides a means by which networked computer systems can agree on a master clock reference time, and a means by which slave clocks can estimate their offset from master clock time. PTP implementations typically have a clock servo that uses a series of time offset estimates to coordinate the local slave clock with the reference master clock time, a process referred to as clock discipline.

This paper presents our software-only implementation of PTP. Precise time coordination with PTP relies on precise estimates of the send and receive times (time stamps) of messages exchanged between the master and slaves. High precision time stamps can be achieved with the support of specialized hardware interfaces in the physical layer of the network; however, many legacy systems lack such hardware interfaces. A PTP implementation that is not supported by specialized hardware is referred to as a software-only implementation. These implementations must time stamp in higher layers of the network, which introduces large degrees of non-determinism in the time stamp latencies, known as jitter. Achieving precise master-slave time coordination with jittery time stamps is the primary obstacle in the design of software-only PTP implementations.

This paper is organized as follows. Section II is a brief introduction to IEEE 1588 (PTP). Section III introduces PTPd, our open-source, software-only PTP implementation. Section IV provides an overview of clock servo design and the specifics of PTPd's clock servo. Section V presents test results of PTPd's performance in a target application. PTPd achieved precision on the order of microseconds. Section VI presents conclusions, comments on future work, and a link to PTPd's source code.

II. PTP IN BRIEF

A. Masters and Slaves

In PTP, master clocks provide the reference time for one or more slave clocks through the exchange of messages over a network. The protocol determines a unique master among a group of clocks using the Best Master Clock algorithm (BMC). The BMC selects the most stable and accurate clock.

B. Sync Messages

PTP masters send Sync messages. The master records the send time of Sync messages (t_1), and slaves record the receipt time (t_2). The difference between the send and receipt times of Sync messages is the master-to-slave delay (d_{m2s}):

$$d_{m2s} = t_1 - t_2. \quad (2.1)$$

Sync messages are sent once per Sync interval (T_{sync}) (typically 2 s). This makes the master-to-slave delay sampling period (T_{m2s}):

$$T_{m2s} = T_{sync} = 2 \text{ s}. \quad (2.2)$$

C. Delay Request Messages

PTP slaves send Delay Request messages. Slaves record the send time of Delay Request messages (t_3), and the master records the receipt time (t_4). The difference between the send and receipt times of Delay Request messages is the slave-to-master delay (d_{s2m}):

$$d_{s2m} = t_3 - t_4. \quad (2.3)$$

Delay Request messages are sent on intervals uniformly distributed between 2 and 30 Sync intervals. This makes the slave-to-master delay sampling period (T_{s2m}):

$$T_{s2m} = T_{sync} * U[2,30]. \quad (2.4)$$

D. One-Way Delay

PTP calculates an estimate of the message propagation delay. This calculation assumes symmetric propagation delays, so that an average of the master-to-slave and slave-to-master delays cancels the time offset between master and slave. This yields the message propagation delay, which the specification refers to as the one-way delay (d_{prop}):

$$d_{prop} = (d_{m2s} + d_{s2m})/2. \quad (2.5)$$

Assuming symmetric propagation delays is often, but not always, valid. Asymmetric propagation delays cannot be observed by the protocol. They will cause a constant bias in the one-way delay and, in turn, the overall time coordination. The bias will equal half of the magnitude of the delay asymmetry.

Assuming a constant delay asymmetry, an asymmetric delay bias can be eliminated by adding a latency correction to the master-to-slave or slave-to-master delay that cancels the asymmetry; however, assuming constant delay asymmetry also may be invalid.

E. Offset From Master

PTP estimates the time difference between master and slave clocks. This is the master-to-slave delay corrected for message propagation delay, and it is referred to as the offset from master (Δt):

$$\Delta t = d_{m2s} - d_{prop}. \quad (2.6)$$

III. PTPd IN BRIEF

A. Background

The Precision Time Protocol daemon (PTPd) is a software-only PTP implementation. It was developed by two engineering students at Case Western Reserve University over a period of approximately six months as part of an undergraduate senior project.

B. Test and Measurement

PTPd is currently developed for Test and Measurement (T&M) systems. For T&M devices (e.g., volt meters and thermocouple instruments), PTP provides time and frequency coordination for the time-stamping of acquired data, and PTP provides a common time-base for time-triggered data acquisition.

The needs of T&M systems significantly influence the current design of PTPd's clock servo. Most notably, the servo is optimized for the stable network topology typical of test and measurement set-ups.

C. Hardware Constraints

PTPd is a software-only system. It lacks two notable systems found in hardware-supported implementations. First, PTPd uses software time stamps. It records message send and receive times in the software layers of the network stack rather than in the physical layer of the networking hardware (e.g., snooping the MII bus of an Ethernet PHY [2]). Second, PTPd uses a software clock. It adjusts the magnitude of the periodic increment of a time quantity stored in memory. However, PTPd was outfitted with a hardware clock for the tests included in this paper. This was done to allow the clock to be read with minimal jitter by isolating jitter in clock reads from jitter in clock coordination.

PTPd is intended for embedded computer platforms that have minimal computing resources. This includes platforms with sub-100MHz CPUs. The program's CPU utilization is below 1% on a 66 MHz m68k processor, as observed by standard resource utilization monitors like the UNIX `top` utility. Also, PTPd does not require a Floating Point Unit (FPU), or FPU emulation, because it uses only fixed point arithmetic. Efficiency and limitation to fixed-point arithmetic are significant considerations in the design of the clock servo.

D. Software Constraints

PTPd is currently ported to Linux. Most of the PTPd system, including the protocol stack and the clock servo, runs as a background user-space process. This allows PTPd to "play nicely" in typical multi-task computing environments. PTPd relies on simple kernel-space routines for its timely components: the frequency adjustable clock and the message time stamps.

PTPd interfaces with the kernel through standard Linux system calls. Receive time stamps are recorded in the Network Interface Card (NIC) driver, in or close to the receive interrupt handler. The receive time stamps are passed to user-space though an `ioctl()`. The receive time stamp mechanism is included in vanilla (unmodified) Linux version 2.4 and 2.6 kernels. A similar send time stamp mechanism is not included in vanilla Linux kernels, but kernel send time stamps can be added to Linux with only small modifications. The entire modification typically amounts to less than ten lines of code. PTPd can operate acceptably without kernel send time stamps, but it performs better with the lower jitter afforded by kernel send time stamps, especially under heavy CPU loads.

PTPd uses the Linux kernel's software clock along with the `adjtimex()` interface for clock tick-rate adjustment. Linux's clock is an implementation of the hybrid kernel Phase-Locked Loop/Frequency-Locked Loop (PLL/FLL) designed by David Mills for the Network Time Protocol (NTP) project [3]. The interface provides many types of clock adjustments, including a self-tuning PLL servo; however, PTPd uses its own servo loop and relies on only `adjtimex()` frequency adjustment. This combination is effective because the user-space servo is efficient and is not sensitive to execution latency, and `adjtimex()` is accurate and responsive to rate adjustments.

Vanilla Linux is not a real time operating system (RTOS); therefore, it guarantees no bounds on interrupt servicing latencies. Both message receipts and clock ticks are interrupt driven events. Variations in interrupt latencies create jitter in the delay estimates that PTPd uses to coordinate clocks. Jitter presents the greatest challenge to precise time coordination, and it is the most significant consideration in the design of the clock servo.

IV. CLOCK SERVO

A. Overview

Figure 1 is a diagram of PTPd's clock servo. The diagram from left to right shows the data path from the protocol to the clock. The protocol regularly samples the master-to-slave delay (cf. Equation (2.2)), and it intermittently samples the slave-to-master delay (cf. Equation (2.4)). Correspondingly, the offset from master is updated regularly, and the one-way delay is updated intermittently. The figure shows the delay and Sync interval inputs, the offset and one-way delay calculations, the offset and one-way delay filters, and the PI controller that mediates the servo output. The output is a fractional tick-rate adjustment that disciplines the clock.

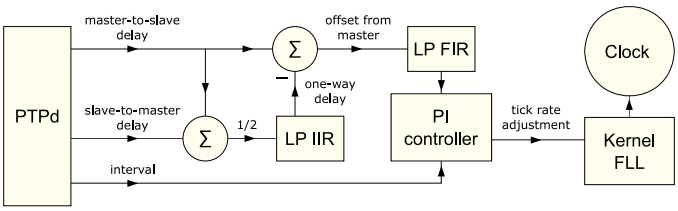


Fig. 1. Clock Servo Diagram

B. Design Parameters

Three characteristics were considered during the design PTPd's clock servo. First is the closed-loop response, including convergence and stability. The acceptable period of initial convergence is on the order of minutes, and the quantity tracked by the servo changes slowly. This allows convergence to be attained and maintained with conservative controller tuning, and conservative tuning largely eliminates stability concerns.

The second characteristic is time error. This represents the time-dependent applications that require two clocks to read the same time at any given point in time. An example of this requirement would be two systems that must take a measurement at precisely the same time. Another example would be two systems that must precisely measure the coincidence in time of two events. A useful metric of time coordination is the root-mean-square (RMS) time difference between clocks.

The third characteristic is rate error. This represents the time-dependent applications that require two clocks to progress at the same rate over a given period of time. An example of this requirement would be a system that measures the frequency content of a signal. It might seem that low rate error must follow implicitly from low time error, but this is not so. A servo design that minimizes time error may sacrifice

rate error, and vice versa. This could occur with an aggressively tuned servo that tracks closely but with a lot of ringing, and the converse case could occur in a sluggishly tuned controller that tracks with a significant offset but is noiseless and steady over short intervals.

A useful metric of rate error over a given period is the modified Allan variance [4, 5] versus summation time (herein referred to as variance versus time scale). The relative tick-rate between clocks typically exhibits three modes of variance: a minimum variance at some medium time scale (typically nanoseconds to many seconds) with increasing variance for small and large time scales. The increasing variance for small time scales represents jitter in the physical oscillator driving the clock. The increasing variance for large time scales represents wander between oscillators caused by changes in the tick-rate due to supply voltage or ambient temperature changes. Clock discipline typically aims to correct oscillator wander and cannot correct oscillator jitter. Ideally, clock discipline should not corrupt the naturally low oscillator variance on medium time scales.

C. Clock Servo Input

The following plots provide a rough picture of the input to PTPd's clock servo. Figure 2 plots PTPd's offset estimate versus master clock time over a roughly one-hour run, and Figure 3 plots the relative tick-rate estimate (the first derivative of the offset in master clock time) versus master clock time. The offset was sampled without PTPd performing any clock discipline. PTPd was a slave to a hardware-supported PTP implementation that achieves sub-microsecond precision. PTPd was running on a 66 MHz m68k embedded Linux platform, with kernel send and receive time stamps.

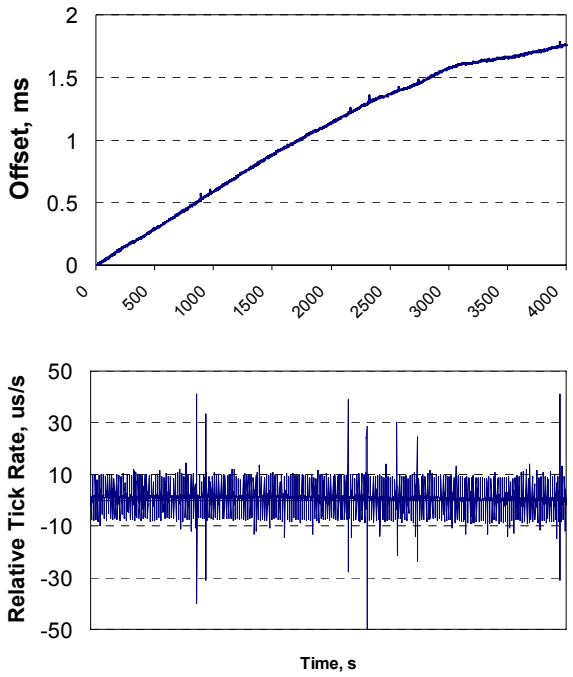


Fig. 2 (top) and Fig. 3 (bottom). Clock Servo Input

The offset signal in Figure 2 is typical of two undisciplined clocks. They drift away from each other in a nominally linear fashion due to inherent tick-rate differences, with some slight curvature due to variations in ambient conditions, including temperature [6].

Figure 3 reveals the microsecond-order noise in the offset signal that is obscured by the large magnitude of the signal in Figure 2. Figure 3 shows two modes of noise. One mode is persistent, high frequency, lower energy noise. Another mode is intermittent, higher energy impulse noise. The noise is not an artifact introduced by the protocol or PTPd because the same modes of noise are exhibited in offsets sampled with an interrupt time stamped master-to-slave pulse-per-second (PPS), one of the simplest means of sampling clock offsets. The persistent noise is likely due to the nominal level of interrupt servicing latency jitter. The impulses may be due to interrupt latencies from extremely long periods of time when interrupts are disabled, or they could be due to periods of delayed execution due to bursty CPU or interrupt loads. Both of these sources of jitter are common in a non-RTOS.

Overall, the noise might appear small because it is orders of magnitude smaller than the long term time loss; however, the 10-30 μs/s noise is orders of magnitude larger than the roughly 0.5 μs/s tick-rate difference that the clock servo must extract from the offset signal to discipline the local clock.

D. PI Controller

The clock servo inputs the offset from master signal into a Proportional-Integral (PI) controller to produce a fractional tick-rate adjustment that coordinates the local clock with master clock time. The PI controller corrects both the time and rate of the local clock. The proportional term tracks and corrects the direct input, which is the time difference between two clocks. The integral term tracks and corrects steady-state error, which is the rate difference between two clocks.

The PI controller approach works well in terms of time error. The controller will drive the time error to zero in stable operation, and there are many analytical tools to optimize PI controller tracking.

The PI controller approach also works fairly well in terms of rate error. The controller tracks just as closely over short intervals as it does over long intervals. This characteristic is effective for correcting oscillator wander, which pushes the Allan variance to zero for long time scales. However, a problem with the PI controller approach arises on medium time scales. The PI controller attenuates noise in its input, but some noise will pass through to its output. This will increase the Allan variance for medium time scales. This problem is often the motivation for windowed and non-linear clock servos [7].

E. Filters

The clock servo uses filtering to mitigate the detrimental effect of input jitter on clock coordination. The filtering attenuates noise in the clock servo input to keep it out of the controller, which keeps jitter out of the clock.

What must be filtered out of the input signal is the persistent noise and the impulse noise described previously. The clock servo uses low-pass filters to attenuate input noise. Low-pass filters are reasonably effective in discriminating between noise and good input. This is because much of the energy in the input signal is close to zero frequency (within the pass-band of a low-pass filter), whereas much of the energy of the input noise is at higher frequencies (within the stop-band of a low-pass filter).

Low-pass filtering is a useful but problematic component of the clock servo. Typically, noise does have low frequency energy that can pass through low-pass filters (e.g. impulses, which have an even energy distribution throughout the frequency spectrum). Lowering the cutoff of the filter attenuates more noise, but lower cutoffs incur greater filtering delays. Delays make the controller less responsive to wander, which increases the tracking error.

Another problem is that low-pass filters can be biased by

colored noise. This could be caused by asymmetric jitter, and would result in a constant offset in the clock coordination. Such biases are typically not a problem because, as previously described, constant offsets can be zeroed by adding a latency correction to the master-to-slave or slave-to-master delay.

The clock servo filters both the offset from master and the one-way delay. The offset from master filtering is only a simple, two-sample average:

$$y[n] = x[n]/2 + x[n-1]/2. \qquad (4.1)$$

This is a Finite Impulse Response (FIR) low-pass filter with a rather high cutoff near the Nyquist rate, but it has minimal delay. This filter effectively attenuates high frequency noise, which the controller does not attenuate as effectively. The one-sample delay incurred through the filter introduces negligible tracking error.

The one-way delay filtering is more involved than the offset filtering. The one-way delay filter is a variable cutoff/phase, first-order Infinite Impulse Response (IIR) filter:

$$s*y[n] - (s-1)*y[n-1] = x[n]/2 + x[n-1]/2. \qquad (4.2)$$

Figure 4 shows the one-way delay filter's frequency response, plotted from zero to the Nyquist rate.

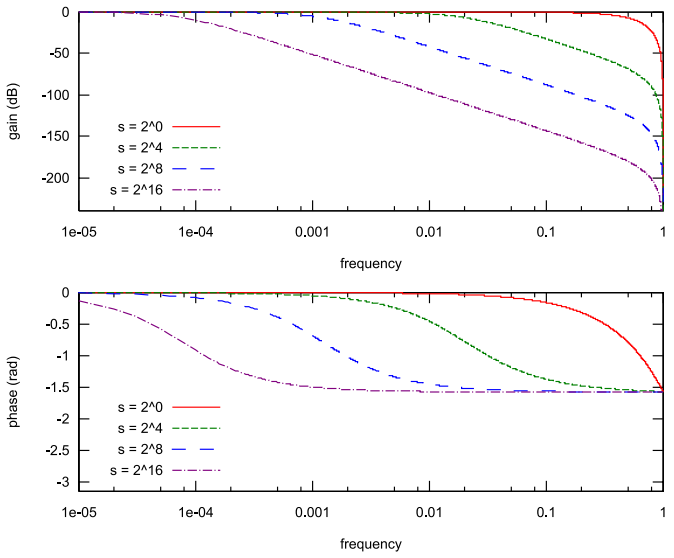


Fig. 4. Frequency Response of Equation (4.2).

For those that are more comfortable with statistical analysis than signal processing, the one-way delay filter can be viewed as a modified exponential smoothing calculation. The standard exponential smoothing form is modified for fixed point arithmetic, and a two-sample average is added to improve the response characteristics at high frequencies.

The 's' term in Equation (4.2) controls the cutoff and phase of the filter, and the term is herein referred to as the stiffness. With a stiffness of one, recursion is eliminated, leaving only a two-sample average (a low-pass FIR filter). Increasing the stiffness lowers the cutoff, but increases the delay.

The clock servo uses this variable cutoff/phase to allow the filter to overcome initial filtering delays at start-up. The servo begins with a stiffness of one, and then increments the stiffness each sample until reaching some maximum stiffness. As the stiffness is increased, the filter cutoff is lowered, and the one-way delay signal becomes smoother.

PTPd's clock servo filters the one-way delay separately from the offset from master. This is for two reasons. The first reason is that the one-way delay signal has a lower nominal sample rate than the offset signal (cf. Equation (2.1-2.6)). The one-way delay signal is therefore interpolated in the combined offset from master signal. This interpolation lowers the frequency of the one-way delay noise, which pushes more noise into the pass-band of the low-pass filters. Filtering the one-way delay directly eliminates the interpolation seen by the filter.

The second reason why the one-way delay is filtered separately is due to the one-way delay signal's having different characteristics than the offset signal. The one-way delay signal reflects the message propagation delay, and its characteristics depend upon the network topology. In the case of the typical T&M set-up, the one way delay is nominally close to constant. A constant one-way delay signal can be filtered through a low-cutoff, high-phase, low-pass filter without increasing the tracking error of the clock servo. This is because there is no time delay of a constant signal through a real filter.

Some applications may not offer a stable network topology; therefore, the one-way delay signal would not be nominally a constant. The current filtering scheme in PTPd's clock servo may not be appropriate for such applications. However, the general approach of treating the one-way delay separately from the offset from master would remain a useful approach.

V. TESTS

A. *Test Set-up*

PTPd is currently being developed for the VXI Technology EX1048 precision thermocouple instrument [8]. The EX1048 is a 66MHz m68k embedded Linux platform. The following tests exhibit PTPd running as a slave connected over an Ethernet hub (except where noted) to a hardware-supported master clock. The EX1048 was coordinated with a hardware-supported master clock because it is expected that a T&M set-up coordinated with IEEE 1588 will include a hardware supported master clock. The master clock for the test is an Agilent LXI IEEE-1588 Demonstration Kit. It is a non-production device made available to the LAN Extensions for Instrumentation (LXI) Consortium for IEEE 1588 testing. Information on the LXI Consortium is available at [9].

The Linux kernel receive time stamps are used, and the kernel is modified to add kernel send time stamps. The Linux software clock is replaced by a frequency adjustable hardware clock implemented in an FPGA. The hardware clock is able to latch the time of received pulses with sub-microsecond precision, but the time is recorded with only microsecond quantization. This is sufficient to test PTPd's coordination, which is on the order of microseconds. Again, the hardware clock is used to allow the clock to be read with negligible jitter by isolating jitter in the clock from jitter in the observation.

The clock coordination is observed by the slave clock recording the time of pulses-per-second (PPS) generated by the master clock. This yields a 1Hz sampling of the slave clock's time with respect to the master clock.

B. *Filtering*

Figure 5 shows time offset between master and slave with various levels of filtering in the clock servo. The top run shows the results of sending unfiltered input to the PI controller. The jitter in the input makes it through to the clock and results in a poor time base. The low frequency undulations are likely due to the large impulses in the input, and the higher

frequency noise on top of the undulations is likely due to the persistent noise in the input.

The middle run uses the fully configured clock servo with filters, but with a one-way delay filter stiffness of one (equivalent to a two-sample average). The offset from master is also filtered by a two-sample simple average. The high frequency noise appears slightly smoother, and the undulations seem slightly smoother as well.

The bottom run has a one-way delay filter stiffness of 2^6, and the coordination is significantly smoother. Most notably, the large undulations have been cut down to small intermittent excursions. These excursions are likely due to impulse noise in the input that is not fully attenuated in the clock servo.

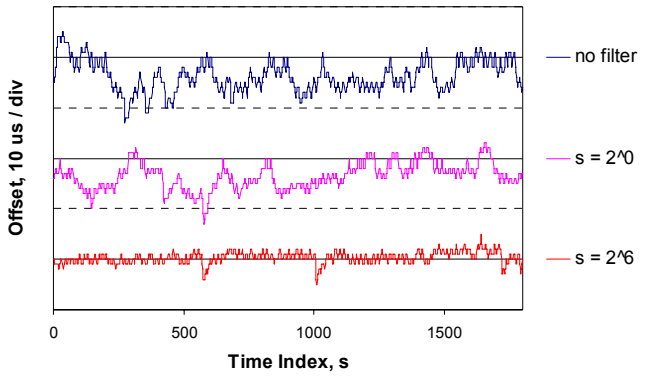


Fig. 5. Filtering Test

C. *Convergence*

Figure 6 shows the time offset between master and slave during the first ten minutes after PTPd starts-up and performs an initial clock reset. Figure 7 shows the next roughly hour-and-a-half after the initial convergence period.

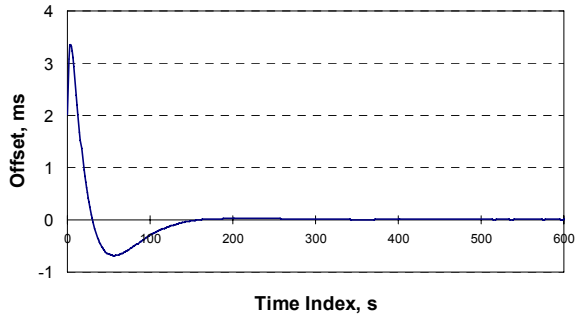


Fig. 6. Convergence Test, 0-10min

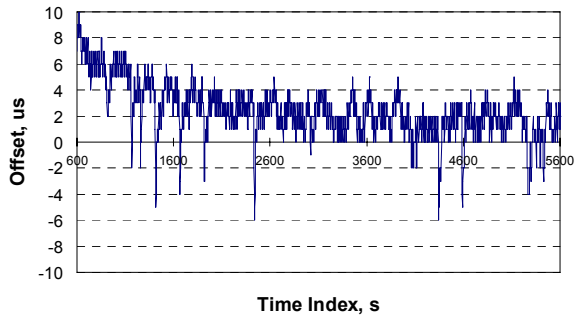
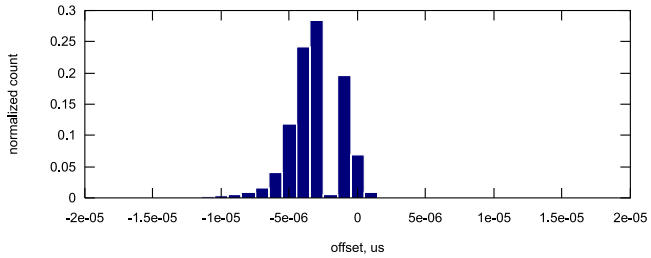


Fig. 7. Convergence Test, 10-90min

Figures 6-7 show that coordination is within ~100 µs after roughly two minutes, and it is within 10 µs after roughly ten minutes. The response characteristics of the PI controller dominate the initial convergence because the one-way delay filter has low stiffness values during this period. The servo does not fully converge for about an hour. During this fine convergence period, the one-way delay filter stiffness is increasing and the filtering delay of the one-way delay signal dominates the convergence.

D. *Precision*

Figures 8-9 shows two histograms of the time offset between master and slave after the clock servo is well converged. The histograms contain 1 µs bins with 50,000 offset samples at 1 Hz (almost fourteen hours). Figure 8 is from a test in which the slave was connected to the master through an Ethernet hub, and Figure 9 is from a test in which the slave was connected to the master through an Ethernet switch.



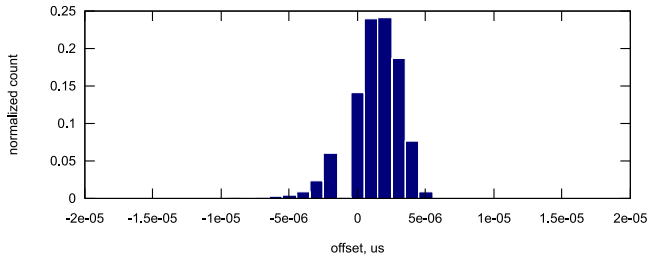


Fig. 8 (top) and Fig. 9 (bottom). Offset Histograms

The histograms show that the offset distributions for both runs are within 10 µs. The offset distribution of the switch run is nearly as tight as the hub run. This indicates that jitter due to switch queuing is insignificant with respect to the slave's internal jitter. There is a bias in both of the distributions, but this is not a concern because the tight distribution indicates that the bias is stable; therefore, it can be eliminated with a latency correction as described previously.

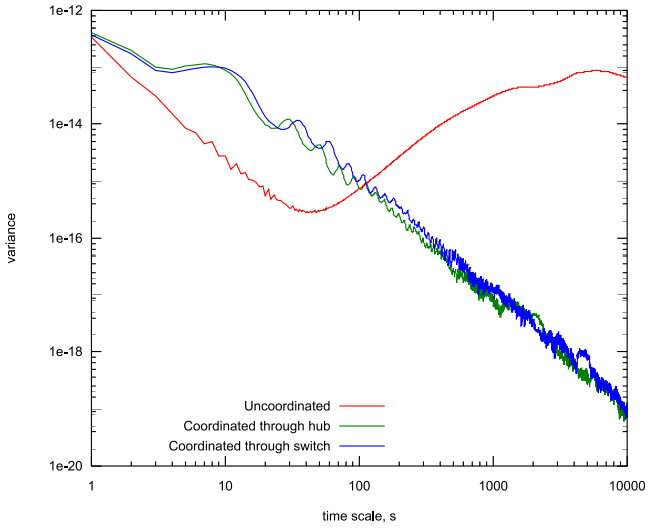


Fig. 10. Allan Variances

Figure 10 shows Allan variance plots of the same 50,000 sample runs versus the variance of an uncoordinated run. The variances for the coordinated hub and switch runs are nearly on top of each other in the plot. The uncoordinated variance has the V-shape typical of uncoordinated clocks due to oscillator jitter on small time scales, a naturally low oscillator variance on medium time-scales, and oscillator wander on large time scales.

Figure 10 shows the advantage of using a PI controller. The variance of the coordinated clock goes to zero for large time scales. This indicates that the clock servo is properly correcting the wander between the oscillators, and it is a result of stable controller tracking.

Figure 10 also shows the troubles with PI controllers. The coordinated clock's variance on medium time scales (1-100 seconds shown) is larger than the uncoordinated variance. This is likely due to jitter in the offset estimate passing through the filters, into the PI controller, and in-turn into the clock. The jitter disrupts an oscillator that is naturally smooth on these time scales.

VI. CONCLUSIONS

A. *Performance*

PTPd coordinated the EX1048 with a hardware-supported master clock within 10 µs. This precision comfortably exceeds the needs of the application in which sampling rates will not surpass 1 kHz.

PTPd can fill the needs of applications requiring sub-millisecond precision. PTPd exhibited coordination within ten microseconds on a platform with a slow (66 MHz), fairly busy CPU. It is reasonable to conjecture that PTPd could approach single-microsecond precision on a modern desktop platform with a more powerful (typically multi-gigahertz) CPU running under light CPU loads.

B. *Future Work*

PTPd is currently in the early stages of development. The clock servo is still quite simple and naive. PTPd's clock coordination precision could be increased with improvements to the clock servo design. Most notably, the noisy coordination on medium time-scales could be smoother. This could be addressed with the addition of a non-linear filtering element that could more effectively attenuate impulses in the clock servo input.

There are other improvements that also may be effective. The PI controller could benefit from improved tuning with the aid of formal analytical methods. The controller might also benefit from the use of gain scheduling. Finally, the one-way delay filter could be improved to accommodate unstable network topologies. The addition of some form of dynamic stiffness adjustment would keep the clock servo responsive to changes in the nominal one-way delay.

C. *Open Source*

PTPd is open source software. The source code is available under same BSD-style license as NTP. The project is hosted on SourceForge at ptpd.sourceforge.net.

VII. ACKNOWLEDGEMENTS

Michael Branicky was supported by the NSF (grant CCR-0309910). VXI Technology, Inc. has contributed to the open source community by supporting Kendall Correll and Nick Barendt during PTPd's development. Chad Greenebaum contributed to PTPd's initial development at Case Western Reserve University. Matt McConnell, a Digital Designer at VXI Technology, Inc., implemented the hardware clock used for PTPd's testing.

REFERENCES

1. IEEE Std 1588-2002. (for more information see http://ieee1588.nist.gov/)

2. Weibel, H., Béchaz, D., "IEEE 1588 Implementation and Performance of Time Stamping Techniques." *Proceedings of the 2004 Conference on IEEE 1588*, 2004.

3. Mills, D.L., and P.-H. Kamp. "The Nanokernel." *Proceedings of the Precision Time and Time Interval (PTTI) Applications and Planning Meeting*, 2000.

4. Allan, D.W., and J.A. Barnes. "A Modified 'Allan Variance' with Increased Oscillator Characterization Ability." *Proceedings of the 35th Annual Frequency Control Symposium*, 470-475, 1981.

5. Allan, D.W., N. Ashby, and C.C. Hodge. "The Science of Timekeeping." Hewlett Packard Application Note 1289, 1997.

6. Allan, D.W., et. al. "Precision Oscillators: Dependence of Frequency on Temperature, Humidity and Pressure." *Proceedings of the 1992 IEEE Frequency Control Symposium*, Report of Working Group 3 of the IEEE SCC27 Committee, 1992.

7. Veitch, D., Babu, S., Pàsztor, A., "Robust Synchronization of Software Clocks Across the Internet." *Internet Measurement Conference*, 2004.

8. VXI Technology, Inc., http://www.vxitech.com/

9. LXI Consortium, http://www.lxistandard.org/

Servo Design for Software-Only IEEE 1588

Kendall Correll, VXI Technology, Inc.
Nick Barendt, VXI Technology, Inc.
Michael S. Branicky, EECS, Case Western Reserve Univ.

Overview

- PTPd is our software-only implementation
  - Still in early stages
- Topics for this presentation
  - Overall Design
  - Clock Servo Design
  - Tests

Test and Measurement

- PTPd currently developed on T&M systems
- IEEE 1588 included in recent LXI Standard
  - LAN Extensions for Instrumentation

Hardware Constraints

- No hardware time stamping
- Software (or Hardware) Clock
- Minimal CPU (no FPU, <100MHz)
- Legacy Systems

Software Constraints

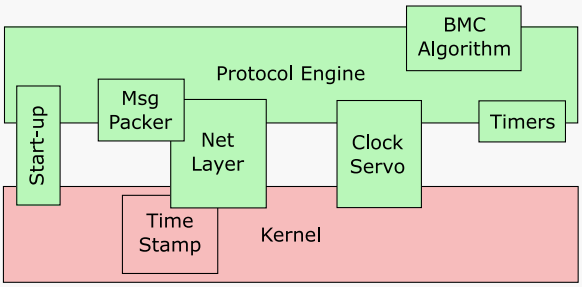
- Ported to Linux (non-RTOS)
- Background User-Space Process
- Kernel Interfaces
  - ioctl() Rx interrupt time stamp
  - Optional custom ioctl() Tx time stamp
  - adjtimex() clock adjustment (NTP)

Code Stats

- ~3200 lines of C, ~100 for servo
- Fixed point arithmetic
- ~44KB x86 binary, ~2KB heap
- <1% CPU usage on 66MHz processor
- Runs on vanilla Linux kernels
- Coded by two undergrads in 6 mos.

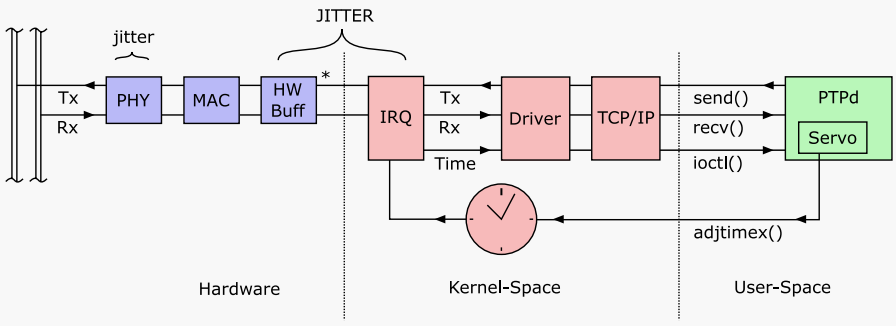
Code Organization

Diagram of PTPd's Major Logical Components



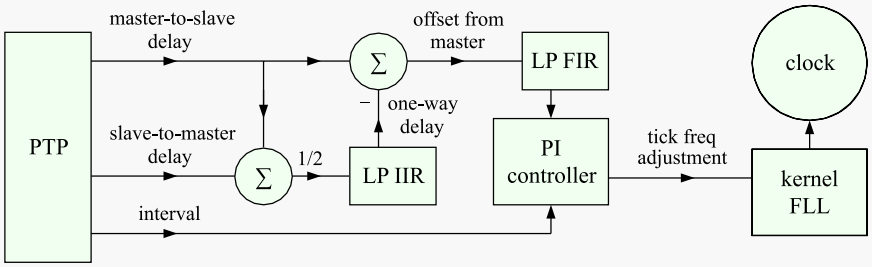
Servo Design

Time Dependent Data Paths



Servo Design

PTPd's Clock Servo

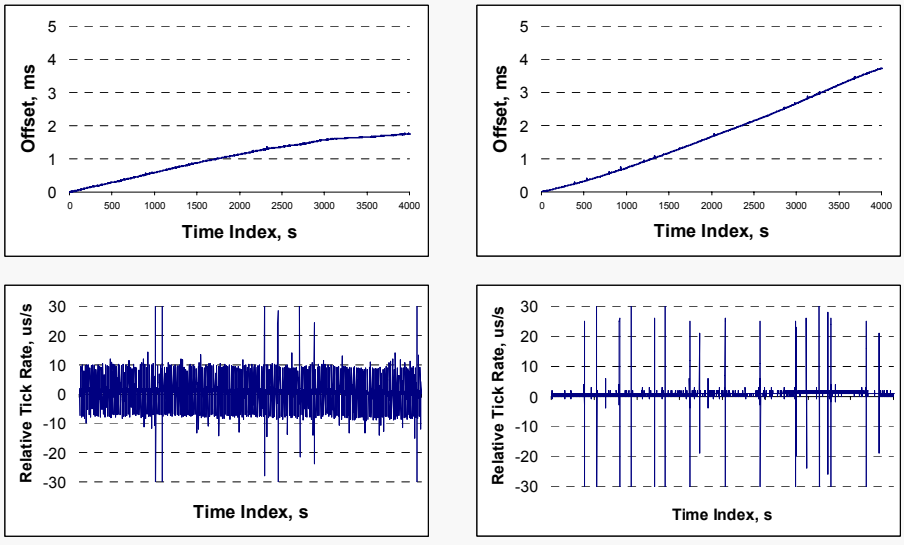


Servo Design - Parameters

- Convergence
  - P & I tuning, filter cutoff/delay
- Time Error
  - Characterized by small RMS error
- Rate Error
  - Characterized by Allan variance
    - Variance -> 0 for large time scales (Tau)
    - Keep naturally low variance for medium Tau
    - Cannot correct increasing var. for small Tau

Servo Design – Input

- Servo input is the offset-from-master
- Components of offset calculation have different sampling period
  - Master-to-slave delay
    Ts = (sync interval)
  - Slave-to-master delay
    Ts = (sync interval)*U[2,30]



PTPd's offset estimate vs. PPS offset estimate

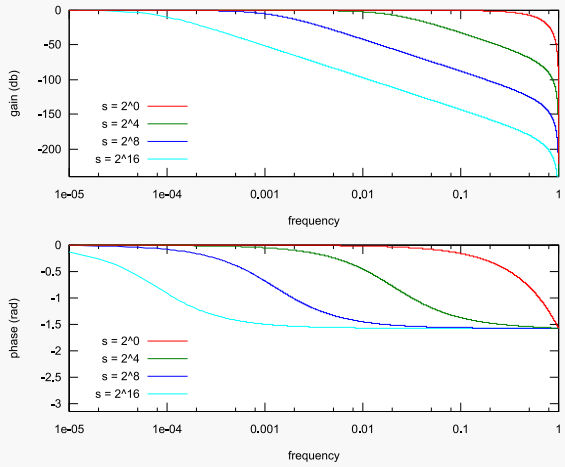
Servo Design – PI

- P tracks input, corrects time difference
- I tracks s.s. error, corrects rate difference
- Advantages
  - Smooth, tunable tracking
  - Pushes Allan variance to 0 for large Tau
- Disadvantage
  - Controller tracks jitter which corrupts clock
  - *Increases* Allan variance for medium Tau
  - Mitigated with filtering

Servo Design - Filtering

- Why
  - Keep jitter out of controller / clock
  - Persistent low energy jitter
  - Periodic high energy impulse jitter
- Where
  - One-way delay, nominally constant
  - Offset from master, nominally a ramp
- How
  - Low-Pass filters

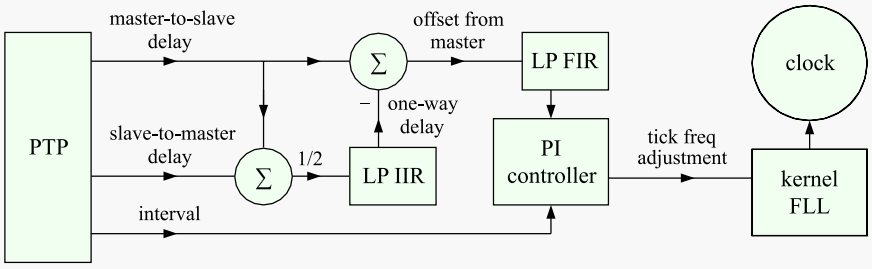
Freq Response of:
s*y[n] - (s-1)*y[n-1] = x[n]/2 + x[n-1]/2



(S = 1 : FIR, S > 1 : IIR)

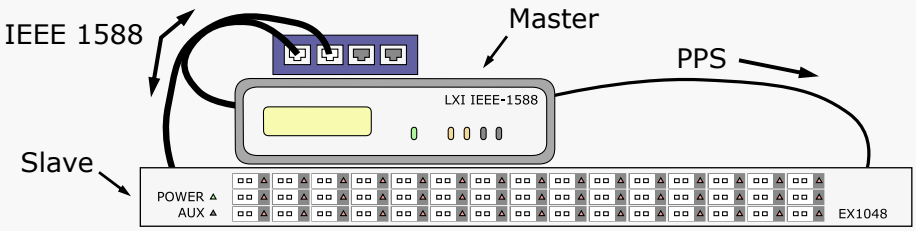
Servo Design – Recap

PTPd's Clock Servo



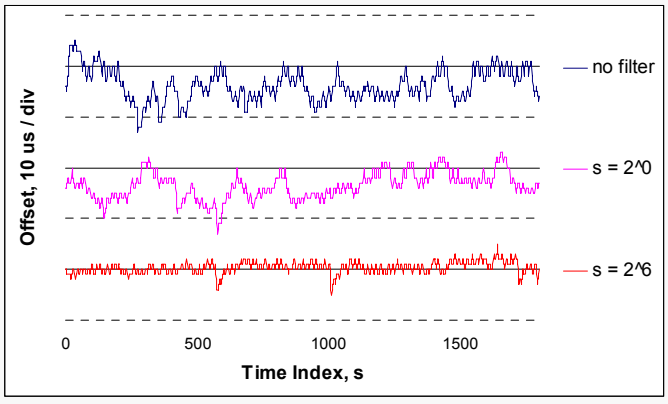
Test - Setup

- Master w/ Hardware Clock & Time Stamp
- Software-only Slaves w/ Hardware Clock
- Time Offset Measured with PPS
  - PPS sent by master, time stamped by slave

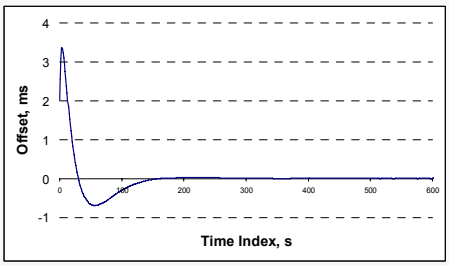


Test – Filtering

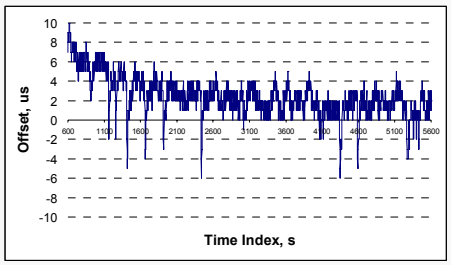
Coordination smoothed w/ filtering



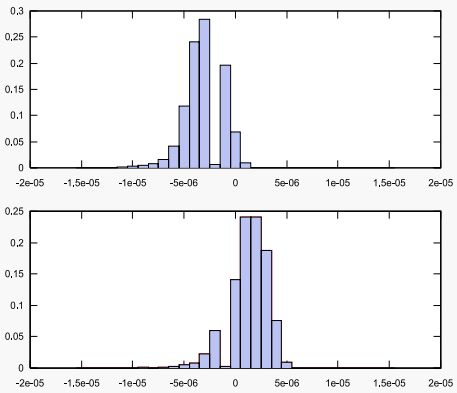
Test – Convergence



Slave Offset
From start-10min

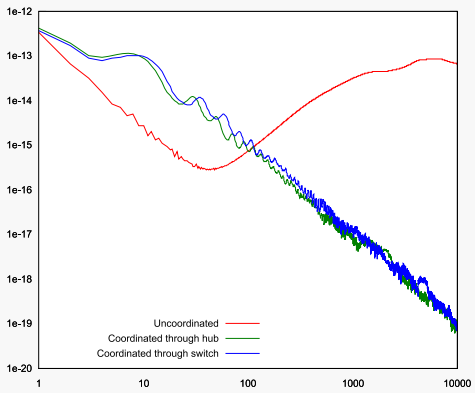


Slave Offset
From 10-90min



Time Offset Histograms,
Ts = 1s, 50 000 samp.
1 µs bins

Coordinated w/
Hub (top), Switch (bot.)



Allan Variances,
Ts = 1s, 50 000 samp.
1 – 10 000 sec

Uncoordinated vs.
Coordinated w/
Hub, and Switch

Conclusion

- Clock Servo
  - Converges <100us in ~2min
  - Coordination <10us
- Overall
  - Software-only
  - Light-weight
  - Free

Check out the code

- PTPd is Open Source
- BSD-Style License
  - Free to use, modify, distribute

ptpd.sourceforge.net

Credits

- VXI Technology, Inc.
- NSF grant CNS-0329910 (Branicky)
- Chad Greenebaum
- Matt McConnell
  - Digital Designer, VXI Technology
  - Various development work, including the hardware clock

Investigation of IEEE 1588 on Gigabit Ethernet, Priority Tagged Frames and Ethernet Daisy Chain

Sivaram Balasubramanian, Anatoly Moldovansky and Kendal R. Harris
Rockwell Automation

Abstract

This paper presents details of an experimental implementation of IEEE 1588 on gigabit Ethernet and its results. Further, it presents experimental frame format for IEEE 1588 messages to facilitate quality of service for network traffic through Ethernet priority tagged frames. Finally, it also presents details of experimental implementation of transparent clocks in Ethernet daisy chain. The transparent clocks are realized through a correction field added to the end of Sync, Follow_Up, Delay_Request and Delay_Response messages and through minor changes to the behavior of end nodes.

1.0 Introduction

Gigabit Ethernet represents the next major evolution of 100Mbps Ethernet and there has been increasing interest in its deployment for industrial applications. Hence it is pertinent to explore IEEE 1588 implementation on gigabit Ethernet. We present the results of experimental implementation of IEEE 1588 on gigabit Ethernet in following sub-section. Quality of service in network transmissions is an important requirement for some industrial applications. Quality of service can be realized through IEEE 802.3 priority tagged frames. A mapping for PTP messages over priority tagged frames doesn't exist in current 1588 standard. We present a simple mapping of PTP messages over IEEE 802.3 priority tagged frames. Full duplex Ethernet daisy chain is a naturally preferred topology for some industrial applications. Synchronization accuracy is a major concern in such linear chain networks. Transparent clock provides a mechanism to realize higher synchronization accuracy on daisy chain networks. We present a simple hardware implementation of end-to-end transparent clock to facilitate full duplex Ethernet daisy chain.

2.0 Gigabit Ethernet

Gigabit Ethernet over copper uses full duplex baseband transmission over four pairs of category 5 balanced cabling. Each pair achieves data rate of 250Mbps towards an aggregate of 1000Mbps over four pairs. The use of special circuits such as hybrids and cancellers enables full duplex transmission by allowing symbols to be transmitted and received on the same wire pairs at the same time. Baseband signaling with a modulation rate of 125 Mbaud is used on each of the wire pairs, which is same as 100Mbps Ethernet.

Four dimensional, five-level Pulse Amplitude Modulation (4D-PAM5) is employed for transmission over four wire pairs. For transmission, 8-bits of data is encoded using 8B1Q4 scheme into a 4-tuple (a, b, c, d) of one dimensional quinary symbols taken from the set {2, 1, 0, -1, -2}. Each of the quinary symbols is then mapped to a unique voltage signaling level of five-level PAM5 for transmission over each wire pair. On receiving side a reverse process is employed to recover data.

Gigabit Ethernet uses 125MHz GMII bus interface for communication between MAC and PHY. The GMII bus includes independent 8-bit data bus each for transmitted and received data, along with independent transmit and receive control and clock signals.

The gigabit Ethernet prototype board was implemented using 833MHz PowerPC 8540 with two integrated gigabit MAC's. The high speed board uses Broadcom BCM5421 gigabit PHY's and 333MHz DDR SDRAM and. Xilinx Virtex 2 FPGA was used to implement 1588 hardware assist. PCB layout was critical to meet 2ns setup time of GMII signals.

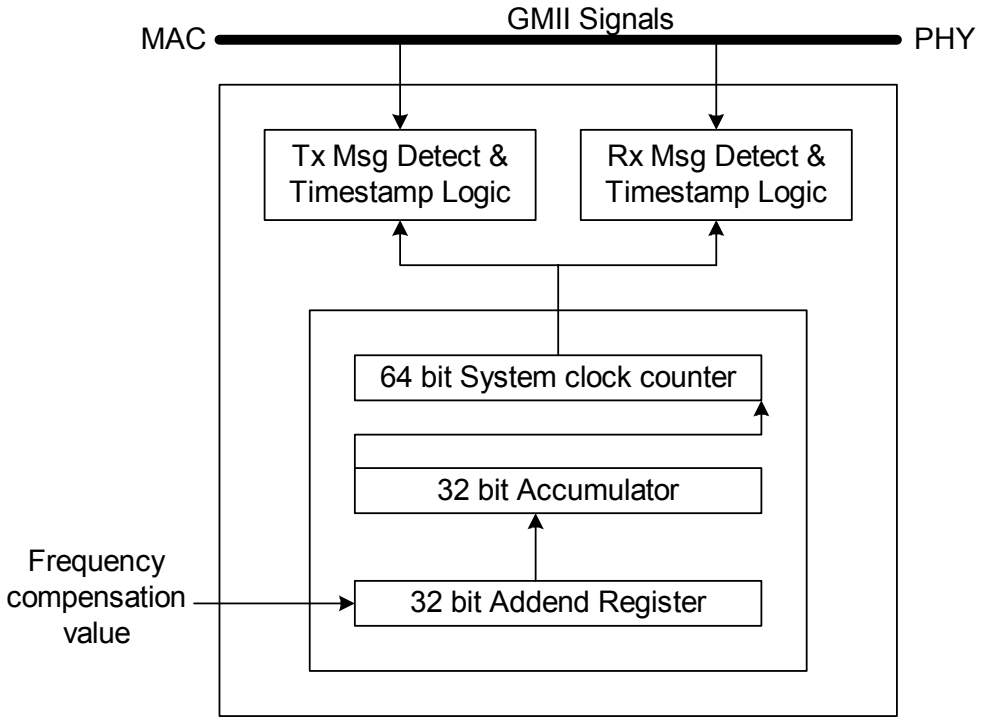


Figure 1: Gigabit Ethernet 1588 Hardware Assist

As shown in Figure 1, the hardware-assist circuit monitors 125MHz GMII transmit and receive signals between MAC and PHY. Two independent PTP message detection logic blocks facilitate time stamping of transmit and receive paths. The hardware assist uses a frequency compensated clock [2] to keep system time. In this circuit a 32-bit accumulator adds the contents of addend register to itself at the frequency of 100MHz. The 64-bit system clock counter is incremented whenever the accumulator overflows. This provides a high precision frequency compensated clock that can be tuned to one part per billion compensation accuracy.

By loading different frequency compensation values to addend register firmware can achieve desired behavior of system time. In this implementation the nominal frequency of system clock counter was 80MHz for a nominal system time resolution of 12.5ns. The firmware implemented 1588 V1.0 ordinary clock with one prototype board serving as time master and another as time slave, inter-connected through an unmanaged gigabit Ethernet switch. The worst case synchronization accuracy was observed to be +/- 50ns. The accuracy

is largely limited by airflow over input oscillator from CPU fan. In future, it should be possible to increase accuracy by restricting air flow over input oscillator.

3.0 Priority Tagged Frames

Figure 2 shows a simple mapping of PTP messages over IEEE 802.3 priority tagged frames. As shown, priority tagged frames have a four byte QTag prefix field following source address and MAC type/length field. The first 2 bytes equals 0x8100 identifying the frame as priority tagged. The subsequent 2 bytes contain frame priority, canonical format indicator and VLAN identifier information.

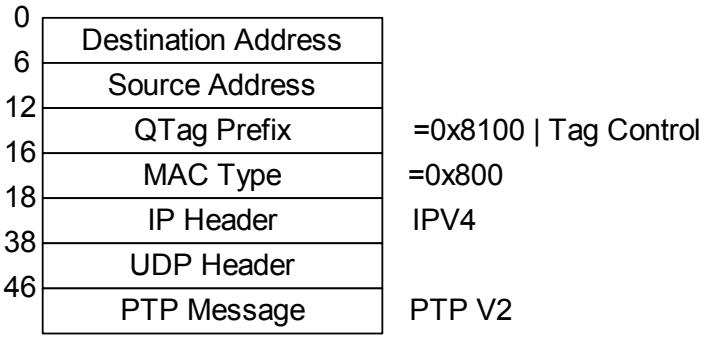


Figure 2: PTP Mapping Over 802.3 Priority Tagged Frame

The proposed PTP mapping doesn't specify a frame priority for PTP messages. However it is expected that most applications will map it to highest frame priority. The proposed PTP mapping follows existing PTP mapping except for addition of QTag prefix field. The proposed mapping was verified through implementation and is described in following sub-section.

4.0 Ethernet Daisy Chain

Figure 3 shows full duplex linear Ethernet daisy chain topology. To achieve this topology a switch with boundary or transparent clock has to be embedded on each node in the chain. Cascading boundary clocks to achieve this topology introduces both accuracy and stability issues. Transparent clocks overcome this limitation by measuring actual residence time for every event message through the switch. With end-to-end transparent clocks the residence times are accounted for ordinary clocks on end nodes resulting in improved accuracy and stability. End-to-end transparent clocks are so called because they don't compensate for path delays, but rely on end nodes to measure propagation delays.

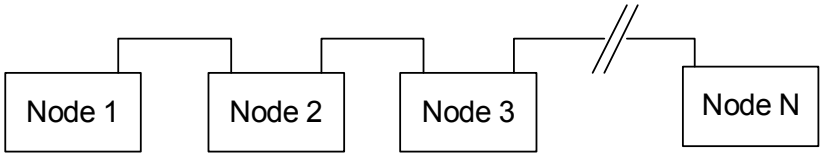


Figure 3: Ethernet Daisy Chain

Figure 4 shows simplified block diagram of a daisy chain node in our implementation. The embedded switch with transparent clock was implemented in Xilinx Spartan 3 FPGA with three ports. Two ports are for daisy chaining and one port is connected to host CPU that implements the functionality of end device with either master or slave clock. The transparent

clock is a hardware only implementation and uses proposed 1588 V2.0 8-byte correction field at the end of PTP messages. The embedded switch supports quality of service through priority tagged frames and uses cut through forwarding technique.

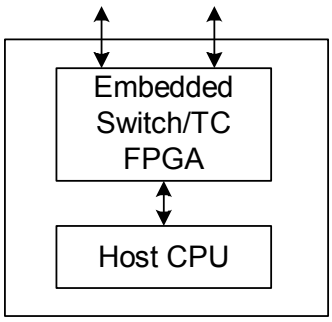


Figure 4: Embedded Switch with Transparent Clock

The transparent clock uses a free running delay timer with resolution of 5ns. When a PTP event message is received on any port, its receive time is time stamped. As that message is transmitted on any other port, its transmit time is time stamped and residence time inside switch is computed. The residence time is added to correction field, message UDP checksum is updated and CRC is updated on-the-fly. When the message reaches end node, the residence times are accounted for in time synchronization computations providing accuracy.

A 33 node daisy chain with one time master node and 32 slave nodes was tested in lab conditions. PTP mapping on priority tagged frames was used with highest priority for time synchronization messages. The network was loaded to 90% of bandwidth using a mix of frame traffic at various priorities. A worst case synchronization accuracy of +/-125ns was observed over several days. The worst case synchronization accuracy exhibited linear trend between first and last slave nodes.

5.0 Conclusion

IEEE 1588 time synchronization on gigabit Ethernet works as expected. A simple mapping of PTP messages over IEEE 802.3 priority tagged frames has been proposed to support quality of service over network transmissions. Time synchronization over full duplex Ethernet daisy chain was realized through transparent clocks. The transparent clocks utilized 8 byte correction field at the end of PTP messages. Both PTP mapping over priority tagged frames and end-to-end transparent clocks work as expected.

6.0 References

1. IEEE 1588, Standard for a precision Clock Synchronization Protocol for Networked Measurement and Control Systems, 2002.
2. Sivaram Balasubramanian, Ken Harris and Anatoly Moldovansky, “A Frequency Compensated Clock for Precision Synchronization using IEEE 1588 Protocol and its Application to Ethernet”, In Proceedings of the Workshop on IEEE 1588, NIST, Gaithersburg, Maryland, Sept. 2003.

LISTEN. THINK. SOLVE.SM

Investigation of IEEE 1588 on Gigabit Ethernet, Priority Tagged Frames and Ethernet Daisy Chain

Sivaram Balasubramanian,
Anatoly Moldovansky and
Ken Harris

ALLEN-BRADLEY • ROCKWELL SOFTWARE • DODGE • RELIANCE ELECTRIC Rockwell Automation

Copyright © 2005 Rockwell Automation, Inc. All rights reserved.

Gigabit Ethernet on Copper

- Uses four pairs of Category 5 balanced cabling
- Each wire pair achieves 250Mbps of full duplex transmitted and received data at the same time
  - Base band signaling with 125Mbaud modulation rate
- Five level Pulse Amplitude Modulation is used over each wire pair
- Uses 8B1Q4 encoding with 4D-PAM5 signaling
  - 8 data bits are encoded to 4 signaling levels for transmission on each wire pair
- Uses GMII interface to MAC
  - 8 bit wide data bus at 125MHz, one each for transmit and receive paths

Copyright © 2005 Rockwell Automation, Inc. All rights reserved.

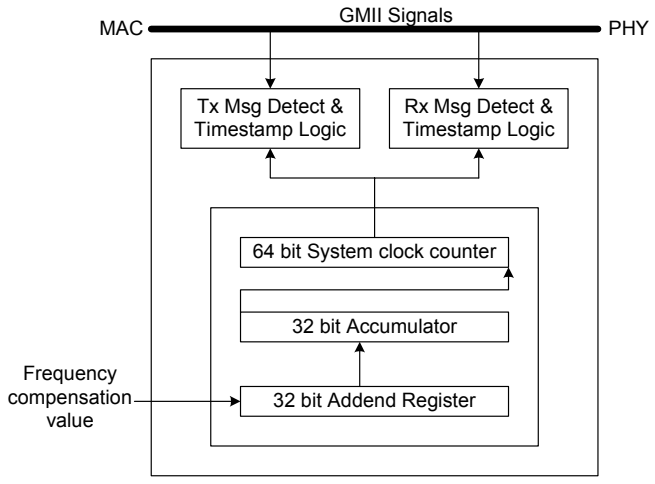
Rockwell Automation

IEEE 1588 Gigabit Ethernet Implementation

- IEEE 1588 V1 Ordinary Clock implementation
- Prototype board is based on the PowerPC 8540 CPU operating at 833MHz with two integrated Ethernet MACs and 333MHz DDR memory
- Uses GMII interface operating at 125MHz
- IEEE 1588 hardware assist circuit was implemented on Xilinx Virtex 2 FPGA
- PCB layout is critical to meet 2ns setup time requirement on GMII interface
- Worst case synchronization accuracy is ±50ns
  - Limited by air flow from CPU fan

Copyright © 2005 Rockwell Automation, Inc. All rights reserved.

Gigabit Ethernet Hardware Assist

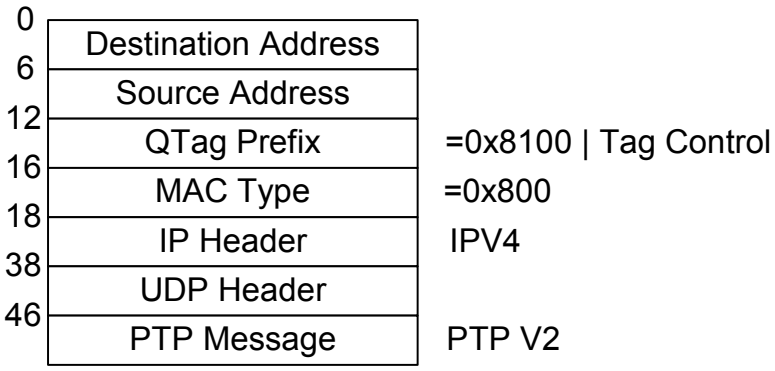


Copyright © 2005 Rockwell Automation, Inc. All rights reserved.

Rockwell Automation

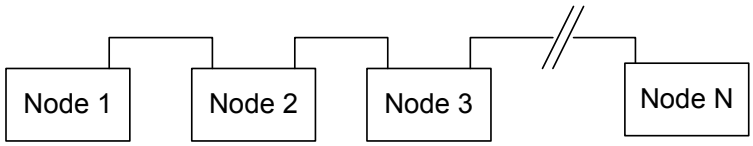
PTP Frame Format for IEEE 802.3 Priority Tagged Frames

- PTP mapping on 802.3 priority tagged frames required for supporting quality of service



Copyright © 2005 Rockwell Automation, Inc. All rights reserved.
5

Ethernet Daisy Chain Topology & Transparent Clocks



- Some Industrial Automation applications prefer linear Daisy Chain topology
- Cascading boundary clocks introduces accuracy and stability problems
- Transparent clocks facilitate linear daisy chain topology
- Transparent clocks measure actual residence time for every 1588 Event message through a node
- Ordinary/Boundary clocks account for residence times in computations resulting in improved accuracy and stability

Copyright © 2005 Rockwell Automation, Inc. All rights reserved.
6

Rockwell Automation

Ethernet Daisy Chain Implementation

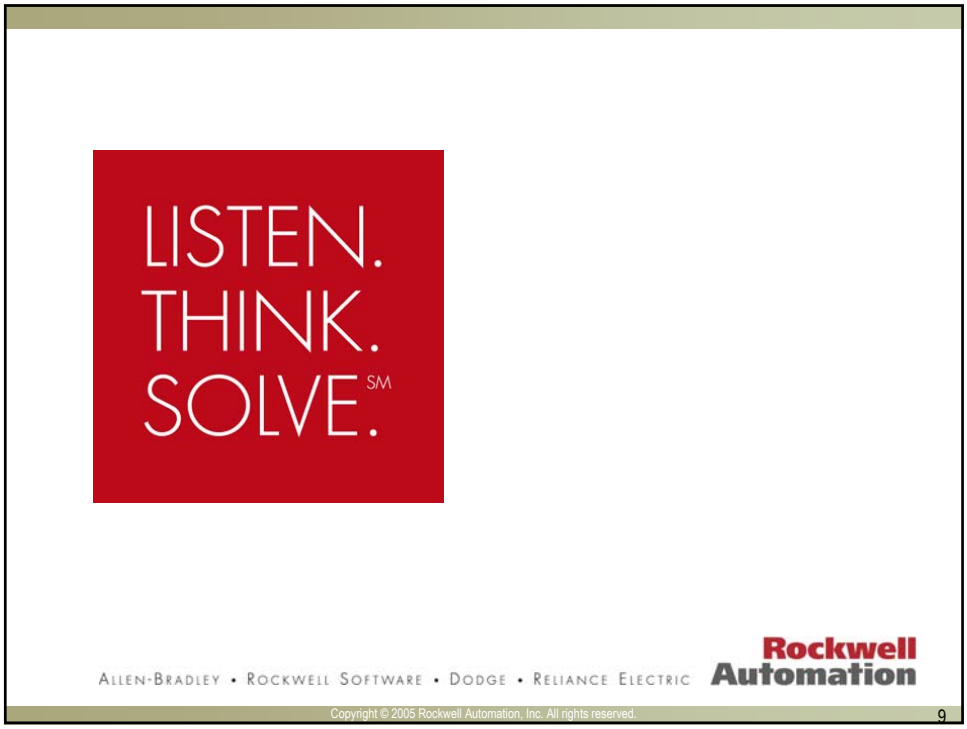
- Each node contains an embedded 3-port switch containing the IEEE 1588 v2 Transparent Clock
- Switch is implemented in a Xilinx Spartan 3 FPGA and is based on the cut-through switching method
- This is a hardware only implementation of the IEEE 1588 V2 End-to-End Transparent Clock which includes:
  - Proposed correction field in the IEEE 1588 messages
  - End-to-End propagation delay measurements
- Proposed IPv4 PTP mappings on Ethernet tagged frames (QoS) are also implemented
- Built a 32-node prototype system in the lab with ±125ns worst case synchronization accuracy on 90% network load

Copyright © 2005 Rockwell Automation, Inc. All rights reserved.

Summary

- IEEE 1588 works on Gigabit Ethernet
- PTP Mapping on 802.3 Priority Tagged Frames facilitates Quality of Service
- Transparent clock facilitates Ethernet daisy chain topology
- Proposed mechanisms for realizing End to End Transparent clocks work

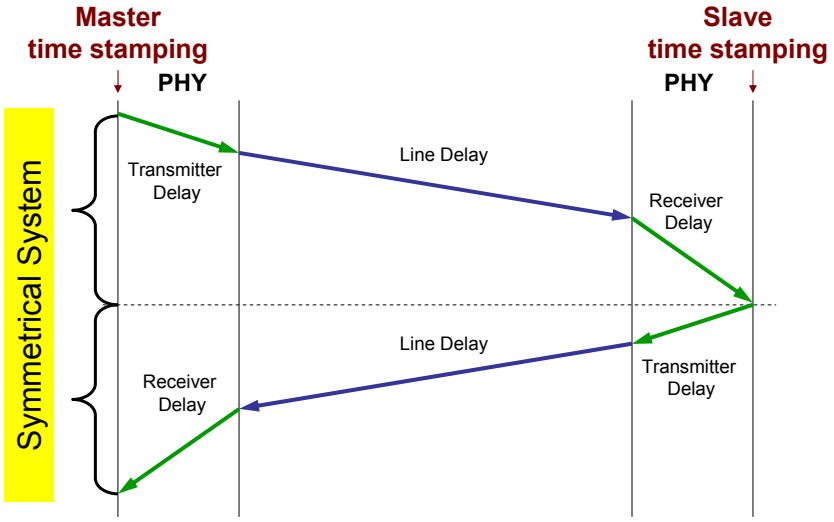
Copyright © 2005 Rockwell Automation, Inc. All rights reserved.


LISTEN.
THINK.
SOLVE.SM
ALLEN-BRADLEY • ROCKWELL SOFTWARE • DODGE • RELIANCE ELECTRIC
Rockwell Automation
Copyright © 2005 Rockwell Automation, Inc. All rights reserved.
9

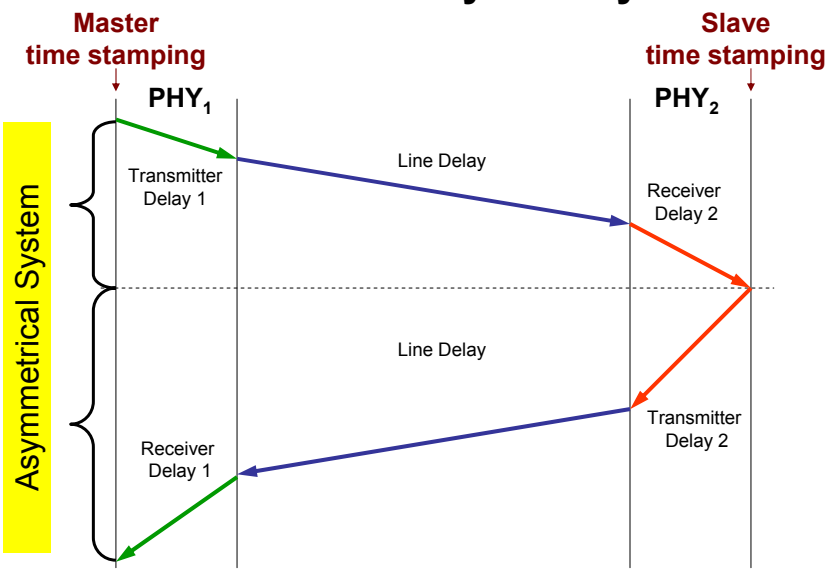
Determination of the IEEE 1588 Relevant Timing Behavior of 100 Base-TX PHYs

Christoph Thurnheer (ZHW)
Jörg Blattner (ZHW)
Prof. Hans Weibel (ZHW)
Dr. Marcel Rupf (ZHW)

Symmetrical System



Problem: Asymmetry



Problem: Receive delay

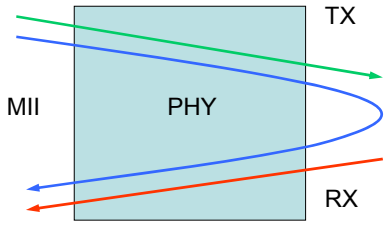
- the receive delay is not always the same

  It is composed of a constant part and a variable part:

  RX Delay = constant part + n*8 ns (n =[0…4])

Goals

- Procedure to measure the delays of a PHY chip

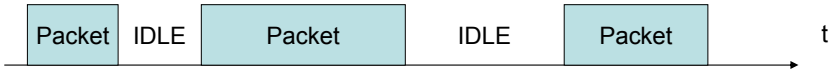


- Compensate the delays for instance in the device driver

How to detect an Ethernet Frame?

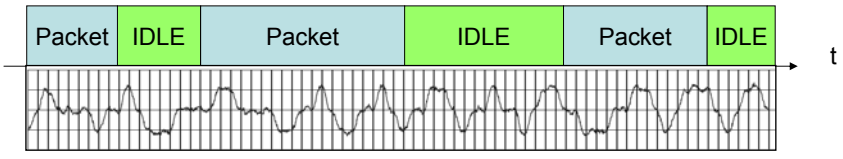
Easy on 10 Base-X Technology:

- Distinct packages with SFD as synchronization point



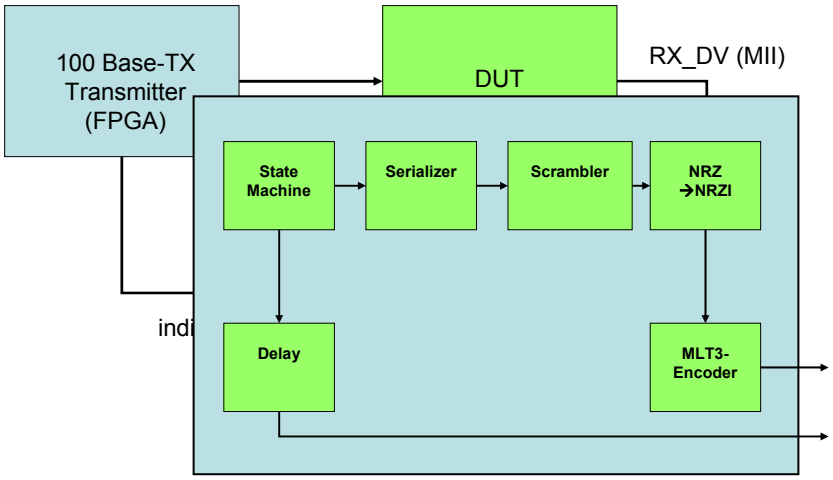
100 Base-TX: Continuous, scrambled Signal

- No visible packages → no synchronization point

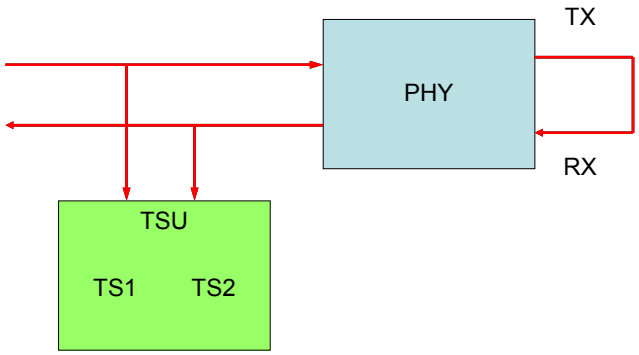


Solution

• for receive delay measurement



• for round trip delay measurement



round trip delay = TS2 – TS1 – TSU internal delays

Results

- We have a working prototype
- All measured PHY chips show the lock problem (RX delay locks into raster of 8 ns after regained synchronization)
- The measured and constant part of the receive delay can be compensated in the device driver

The market will see optimized PHYs

News Release announcing a respective product:

The ….. transceiver addresses the quality and system reliability needs of developers implementing real-time Ethernet in factories and other harsh environments. By removing non-deterministic signaling from the data path, the transceiver ensures timing latency across the media independent interface (MII) and reduced media independent interface (RMII) MAC standard interfaces. ……..

Summary

- the manufacturer have recognized the problem
- The knowledge of the PHYs receive an transmit delay is important but depends on your application

Questions?

Many thanks for your attention!

Christoph Thurnheer (ZHW)

Zurich University of Applied Sciences
Institute of Embedded Systems
http://ines.zhwin.ch

An Update on Nanosecond-level Time-Synchronization with IEEE-1588

Dietrich Vook, Bruce Hamilton, Andrew Fernandez, Jeff Burch, and Vamsi Srikantam

Abstract:

At the 2004 IEEE-1588 conference we reported experimental work on the use of high stability oscillators as a key component in achieving high accuracy synchronization using IEEE 1588 [1,4]. This work has been extended through improving the resolution of the IEEE 1588 clocks. New results obtained on 100 Mb/s Ethernet are presented.

Introduction

Highly accurate time synchronization is of interest in the electronic test and measurement industry as well as telecommunications industry. For the testing of electronic systems, especially RF systems, nanosecond level synchronization is needed. Agilent is a founding member of the LXI consortium (LAN Extension for Instruments) [2]. LXI is an instrumentation platform based on industry standard Ethernet technology designed to provide modularity, flexibility and performance to small- and medium-sized systems. A LXI system may be triggered and synchronized over Ethernet by using the IEEE-1588 standard. Current generation test systems provide these functions by using hardwired triggers and matching calibrated cables. The telecommunications industry requires highly accurate frequency matching to minimize buffer sizes at zone boundaries and accurate time matching for globally synchronizing services [3].

Experimental Setup

To use the IEEE-1588 protocol at the nanosecond level, several system design aspects must be considered. First, the measurement apparatus used to verify the system synchronization must have accuracy well below the nanosecond level. The minimum time resolution of our HP5372A Frequency & Time Interval Analyzer is 200 psec. Figure 1 illustrates our equipment's measurement error. In this experiment, the period of the PPS output (PPS = Pulse-Per-Second) from an Agilent 5071A Cesium atomic clock is measured by a HP5372A. The HP5372A is locked to the atomic clock using the clock's 10 MHz reference. The measured period's deviation from the nominal 1 second is plotted in Figure 1. Equivalent results were obtained measuring the period of the master clock driven by an Agilent 4438C signal generator. Second, to minimize asymmetries in measurements, cable lengths between the Frequency & Time Interval Analyzer and the 1588 Master or Slave units should be matched.

The stability and repeatability of the delays of LAN signals between 1588 nodes is important to the creation of a high accuracy version of the 1588 protocol. With the setup shown in Figure 2, we measured the time delay between transmit-start-of-frame (SOF_{Tx}) and receive-start-of-frame (SOF_{Rx}). The SOF signals are generated in the 25 MHz clock domain driven by the respective PHY clock (TX_{CLK} or RX_{CLK}). This delay includes the PHY delays at both ends as well as the time-of-flight for the signals down the 1 m crossover cable. In Figure 3, you can see that these signals are well behaved with random delays within +/- 0.5 ns. This spread acts like a noise-like dither signal added to the one-way delay of a 1588 packet. The rising edge of the SOF signal is time-stamped by the 1588 circuits at a resolution of 1 ns. These time stamps are averaged by the 1588 servo, reducing quantization issues related to time stamp resolution [5].

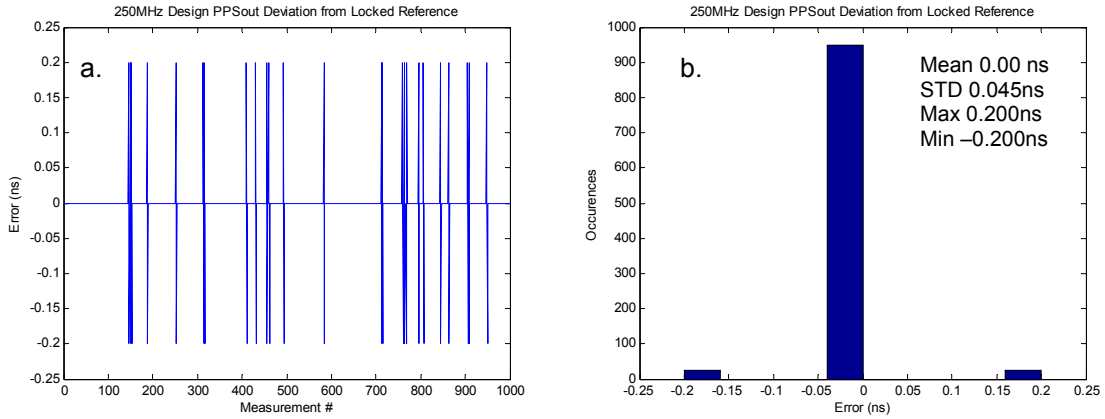


Figure 1, 1000 measurements of the period of an atomic clock locked to a HP5372A a) time series of error in ns vs. measurement number, b) distribution.

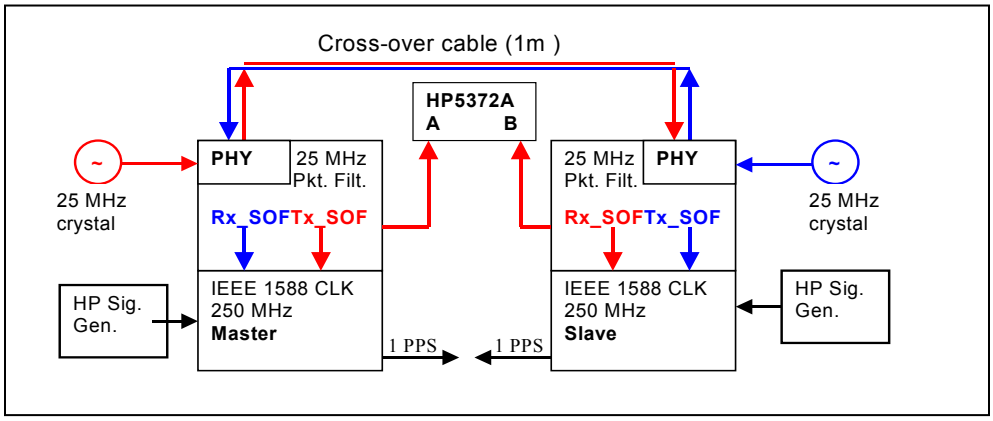


Figure 2, Setup for measuring the SOF_{Tx} to SOF_{Rx} delay.

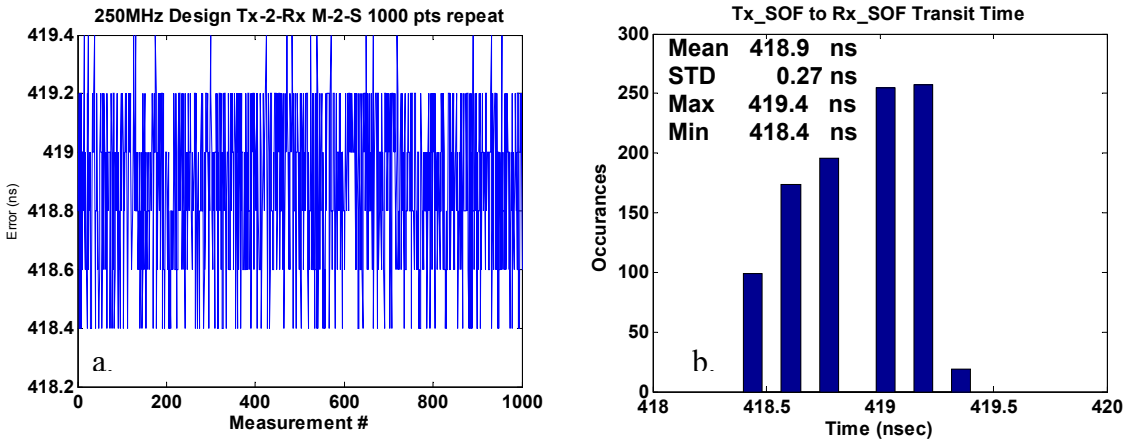


Figure 3, 1000 measurements of SOF_{Tx} to SOF_{Rx} delay over 1 m crossover cable. a) Time series b) histogram of delay times.

Current test systems deal with synchronization and concurrency issues through: a) a hard-wired star trigger topology, b) precisely calibrating cables for length and delay, c) calibrating each subsystem from the internal circuitry to the edge connector, and d) careful system calibration procedures involving locking system oscillators to very stable reference signals, typically 10MHz. One goal of the LXI consortium is to reduce the user complexity by dealing with these triggering and synchronization issues using the IEEE-1588 protocol and system design [2]. For IEEE-1588 based test systems, signal propagation asymmetries will need to be measured and compensated.

This paper focuses on the recent progress toward narrowing the timing distribution. The section below discusses the factors influencing clock-to-clock distributions and system design. The following section shows our clock synchronization distribution results of +/- 2.5 ns over a cross-over cable. The last section discusses the remaining asymmetry issues, why they are important for test and measurement, and related issues that arise in IEEE-1588 systems at the nanosecond level.

Factors Impacting the Width of the Timing Distribution

There are many factors that impact the width of the distribution of time offsets between the 1588 clocks. The ones that relate to the design of the digital accumulators (clocks) are shown schematically in Figure 4. For many implementations, the digital time accumulator is basically a digital adder. Increasing the adder frequency as well as bit depth reduces the quantization errors. Since the 1588 protocol only infrequently adjusts the rate (on the order of seconds) relative to the adder oscillator rate (nanoseconds), one must be able to set this rate very finely. Increasing the resolution (number of bits of the rate representation) allows more fine control of the rate. Alternately, a design could increase the message rate at the expense of network loading. The servo bandwidth must also be optimized. Very narrow servo bandwidths can be used to reduce the spread of the time distribution. This requires more stable oscillators and adaptive control loop (P I) parameters to enable fast servo lock.

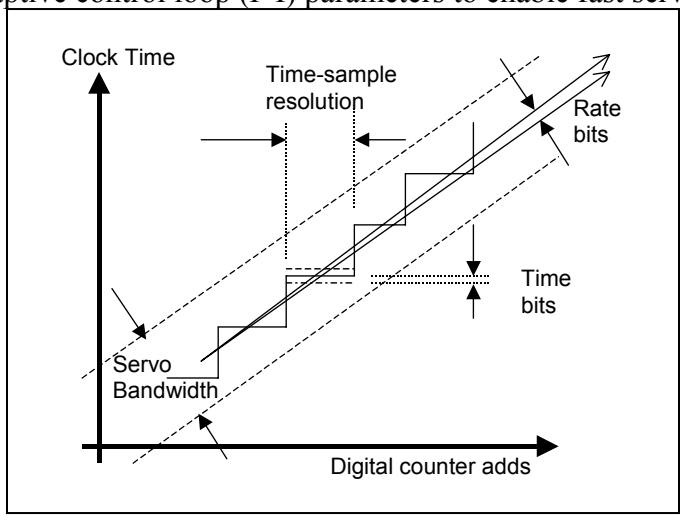


Figure 4 Clock design considerations for the digital clock and how they relate to the distribution of measured clock offsets.

There are other nonstationary factors that broaden the distribution of clock offsets. These include: a) short term oscillator stability, b) variations in message travel time, c) nonstationary issues in the PHY delays, and d) PLL overshoot. We also hypothesize that if the time and rate representations are not carefully matched to the master oscillator frequency, the master 1588 clock will servo, leading to a broadening of the distribution of time offset between master and slave of 1 to 1.3 least significant bits of the underlying time representation in the ideal case. In addition, the finite slew rate of both the LAN and PPS signals leads to a noise sensitivity that can broaden the distribution. The results in Figure 6 below demonstrate that we have minimized most of these issues.

Recent Timing Distribution Results

The clock offsets are measured by time difference between pulse-per-second (PPS) signals. At the IEEE-1588 conference in 2004 we reported a distribution of roughly +/- 12 ns (+/- 3σ) [4]. This result is reproduced in Figure 5 below. This result was obtained using stable, instrument-grade oscillators running at 250 MHz (4 ns period). We have been able to improve upon these results by bringing the effective sample period down to 1 ns. A standard of deviation of 0.771 ns and a spread of +/- 2.5 ns (+/- 3σ) was measured, as seen in Figure 6 below.

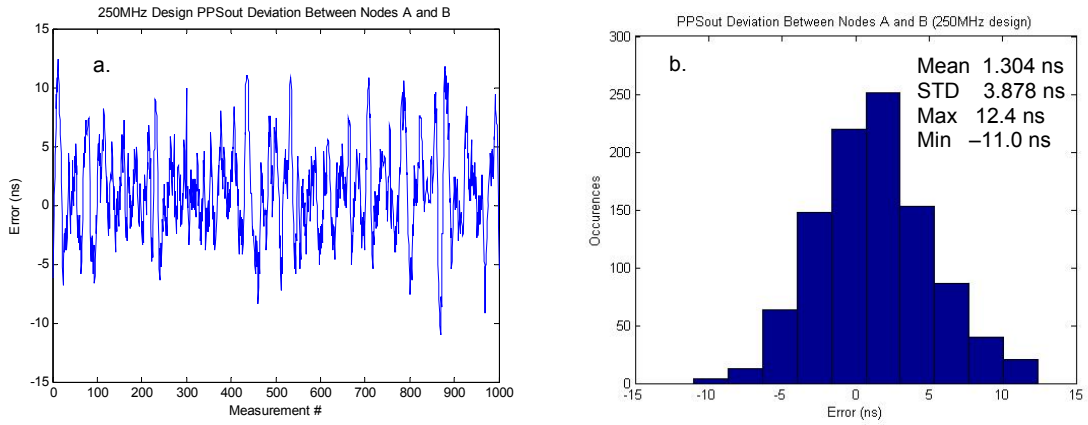


Figure 5, Pulse-per-second time difference reported in 2004 for 4 ns clock period and short cross-over cable, 1000 samples. a) time series, b) distribution.

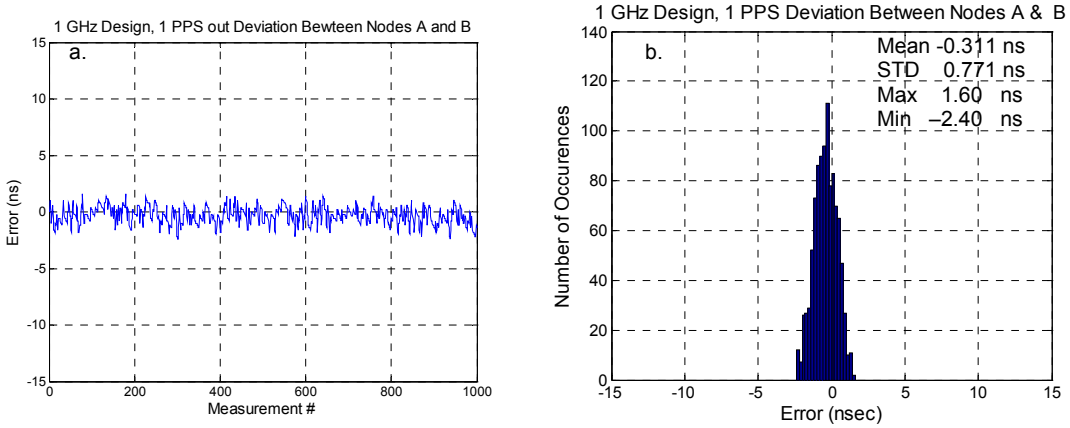


Figure 6, Pulse-per-second time difference for 1 ns clock period and short cross-over cable, 1000 samples. a) time series, b) distribution.

As the sampling time resolution improved over the previous design, we see a significant improvement in the distribution of clock offsets. This is indicated by the reduced standard deviation in Figures 5b and 6b above. A summary of the sources contributing to the width of this distribution is given in Table 1. Thus our designs were dominated by sample rate limitations. This was also true for earlier designs from our laboratory, as shown in Figure 7, where the maximum clock offset between master and slave is shown versus sample resolution. As can also be seen from Figure 5b and 6b, the offsets due to asymmetries are relatively more significant as the distribution is narrowed. This will be discussed in more detail in the next section. We anticipate that the clock-to-clock time synchronization will be within 2.5 to 3 times the minimum time-stamp resolution, as shown in Figure 7. We anticipate improvements in the digital FPGA implementation will improve clock to clock synchronization to +/- 1 ns, and possibly down to the +/- 0.5 ns level in the coming year. We expect a reduction in the distribution to the +/- 100-200 psec level can be achieved using higher accuracy analog circuit techniques. Time-to-digital converter technologies have been in the literature for about 10 years with mixed signal ASICs showing impressive time stamp resolutions in the 25 ps range [6-8]. More recently, FPGA designs with locked placements have yielded impressive results in the 100 to 200 ps range [9-10].

Error Source	Max – Min error (ns)
SOF time-stamp quantization	1
Pulse-per-second quantization	1
Rate quantization over 1 sec	1
PHY jitter	1
1588 Oscillators	< 0.2

Tab le 1: Error sources in offset distribution of Figure 6 b.

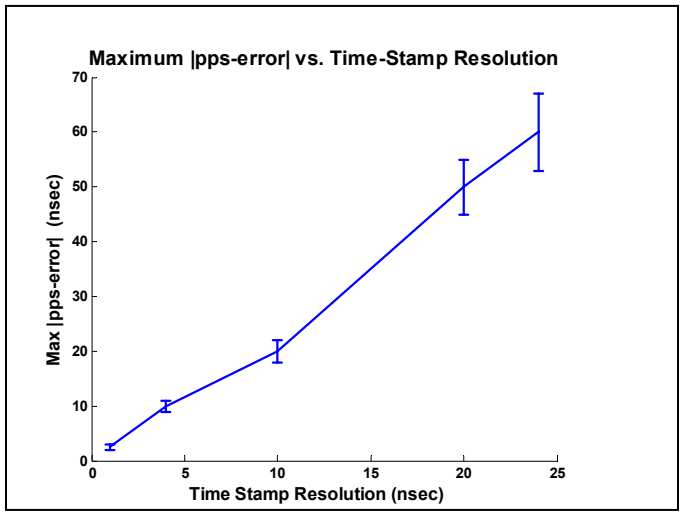


Figure 7, Maximum clock-to-clock difference versus minimum time-stamp resolution, using cross-over cable for various designs from our laboratory.

Asymmetry Issues

The previous section showed how the non-stationary offsets have been minimized through system design. Even if an IEEE-1588 system has a tight time distribution, the protocol only corrects for symmetric latencies. Static asymmetric latencies are not corrected by the protocol. These can arise from many sources, most notably differences in the delay through the PHYs and asymmetries in the LAN cables. These asymmetries become relatively more important as the distribution is narrowed and finer time accuracy is needed.

There are many asymmetries to account for in 1588 systems. The delays between the internal clock circuitry and the PHY will differ. The delays through the PHYs for transmit and receive channels can differ and be time-varying due to start-up issues [12]. General 1588 systems may end up with PHYs from different manufactures, compounding the issue. For this work, the system was constructed with essentially identical PHYs, and these asymmetries cancel for the most part. No start-up asymmetries were observed in our system. Another significant issue is the asymmetric propagation delay down standard LAN cables. This cable skew can be as much as 50 ns for 100 m cables [12].

With our design the cable skew is observable. Using a 60 m LAN cable [11] in series with a short cross-over cable, we can clearly observe the LAN cable asymmetry that is mentioned in the 2004 IEEE-1588 conference [12]. With the 60 m LAN cable put in one way, we get a mean difference between clocks of +6 ns. With the 60 m LAN cable reversed, we get a mean difference of -5.5 ns. These results are seen in Figure 8 below.

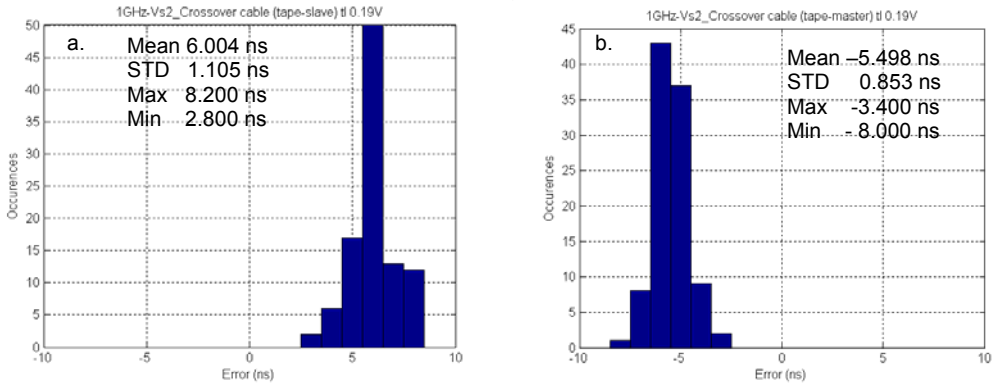


Figure 8, Pulse-per-second time difference for 60 m cable and short cross-over cable, 1000 samples. a) 60 m cable in one direction b) 60 m cable reversed.

Conclusion

We have demonstrated clock distribution below +/-2.5 ns over 1588 using 100Base-T. We expect this accuracy to improve to below 1 ns in FPGA-based designs in the near future and possibly to the 100 ps range if mixed-signal ASICs are produced. We showed that the time offsets due to cable asymmetries are significant when designing systems for this level of accuracy. We infer that other asymmetries in the system will need to be calibrated out to ensure proper system function at the nanosecond level. We list some of these that are readily apparent; others will undoubtedly arise in the process of building systems to this level of accuracy.

Bibliography

[1] http://ieee1588.nist.gov/

[2] http://www.lxistandard.org/home

[3] "IEEE-1588 in Telecommunications Applications," Dave Tonks, Proceedings of 2004 Conference on IEEE-1588 Standard for Clock Synchronization Protocol for Networked Measurement and Control, NIST-TIR 7192, p. 184.

[4] "High Accuracy Clock Synchronization Using IEEE-1588," Pritam Baruah, Pruthvi Chaudhari, Paul Corredoura, John C. Eidson, Andrew Fernandez, Bruce Hamilton, Jeff Burch, John Stratton, Dieter Vook; Proceedings of 2004 Conference on IEEE-1588 Standard for Clock Synchronization Protocol for Networked Measurement and Control, NIST-TIR 7192, p. 128.

[5] "Reducing ADC Quantization Noise: Two Techniques, Over sampling and Dithering," Richard Lions and Randy Yates, Microwaves and RF, June 2005, ED online ID #10586, http://www.mwrf.com/Articles/ArticleID/10586/10586.html

[6] "A High-Speed Wide Dynamic Range Time-to-Digital converter," M. Lampton and R. Raffanti, Rev. Sci. Instrum. Vol. 65, (11) Nov, 1994, p. 3577-3584.

[7] "A Sampling Technique and its CMOS Implementation With 1Gb/s Bandwith and 25ps Resolution," C. Thomas Gray, Wentai, Liu, Wilhelmus A. M Van Noije, Thomas A. Hughes, Jr., Ralph K.l Cavin III, IEEE-Journal of Solid-State Circuits, Vol. 29 (3) March 1994, p. 340-348.

[8] "A High-Resolution CMOS Time-to-Digital Converter Utilizing a Vernier Delay Line," IEEE Transactions on Solid-State Circuits, Vol. 35, (2) Feb. 2000, p. 240-246, Pitor Dudek, Stanislaw Szczepariski, John V. Hatfield, IEEE Transactions on Solid-State Circuits, Vol. 35, No.2 Feb. 2000, P. 240-247.

[9] "Field-Programmable-Gate-Array-Based Time-to-Digital Converter with 200-ps Resolution " Joozef Kalisz, Ryszard Szplet, Jerzy Pasierbinski and Anderzej Poniecki, IEEE Transactions on Instrumentation and Measurement, Vol. 46 (1) Feb. 1997, p. 51-55.

[10] "Interpolating Time Counter with 100ps Resolution on a Single FPGA Device," Ryszard Szplet, Jozewf Kalisz, Rafal Szymanowski, IEEE Transactions on Instrumentation and Measurement, Vol. 49 (4) Aug. 2000, p. 879-883.

[11] Lucent-D Systimax Gigaspeed 1071A CM ED 40651 LAN cable, 60m long.

[12] "PHYs and Symmetrical Propagation Delay," Thomas Müller, Alexander Ockert, Hans Weibel; Proceedings of 2004 Conference on IEEE-1588 Std. for Clock Synchronization Protocol for Networked Meas. and Control, NIST-TIR 7192, p. 78.

Update on
High Precision Time Synchronization

IEEE-1588 Conference and Plug-Fest
Winterthur Switzerland
October 10-12, 2005

Dieter Vook*, Jeff Burch, Andrew Fernandez,
Bruce Hamilton, Vamsi Srikantam

Measurement Research Lab
Agilent Laboratories

Outline

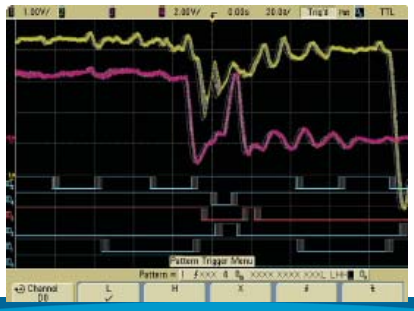
1. Motivation: high precision 1588 applications
2. Project goals
3. Design considerations impacting high accuracy
4. Measurement issues
5. Timing distribution results
6. Asymmetry issues
7. Conclusions

High Precision 1588 Applications

- Electronics test: accuracy needs are application dependent

  Requirements range from μs to sub-ns
  - RF signals analysis (coax: vel. ~ 5 ns/m)
  - SerDes (serial data at > 1 Gb/sec)
  - Networking in GHz rates, packets in low ns

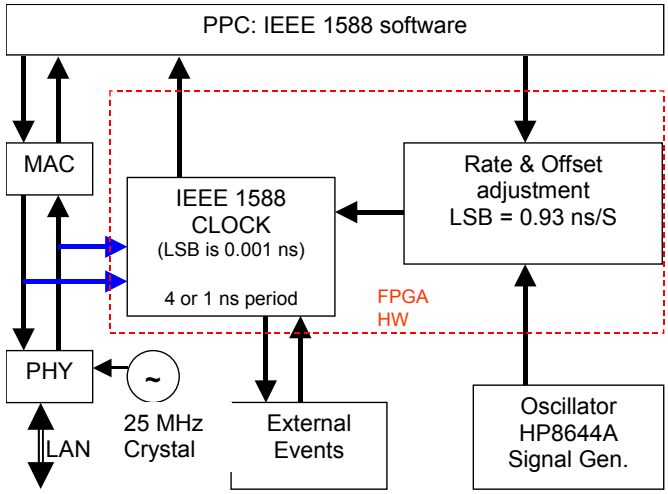
- Other application areas
  - Military (comm. / nav.)
    - Systems with many receivers
    - Synch. over long distances
  - Telecommunications



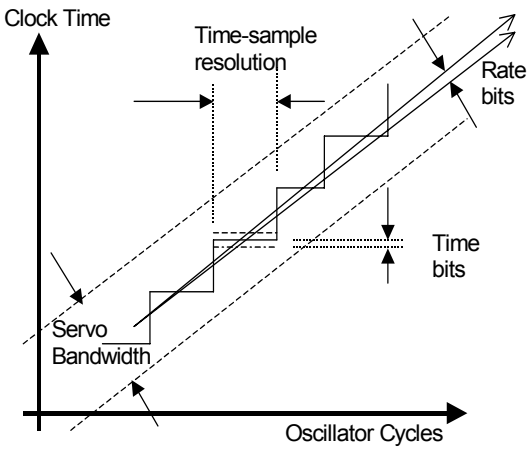
Project Goals

- Extend IEEE-1588 to nanosecond level
  - Stable oscillators
  - 100 Base-T LAN
  - Utilize increased resolution
    - Number of bits: time and rate representations
    - Clock frequency increased to improve time-stamp resolution
- Demonstrate nanosecond feasibility
  - Direct clock-to-clock results (best case)

Experimental Clock Design



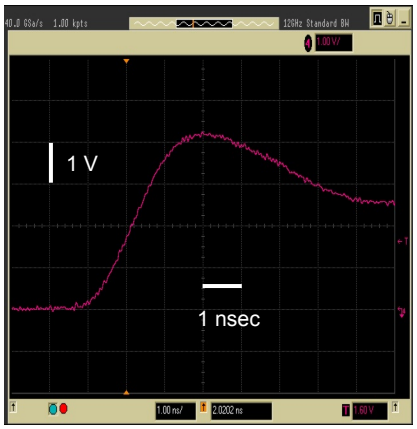
Design Considerations Impacting Clock Accuracy



- Bit depth of time & rate values
- Time-sample resolution
  - → Oscillator freq.
- Narrow servo BW
  - → Oscillator stability
  - → Dynamic P, I

Measurement Issues

At nsec level many things matter

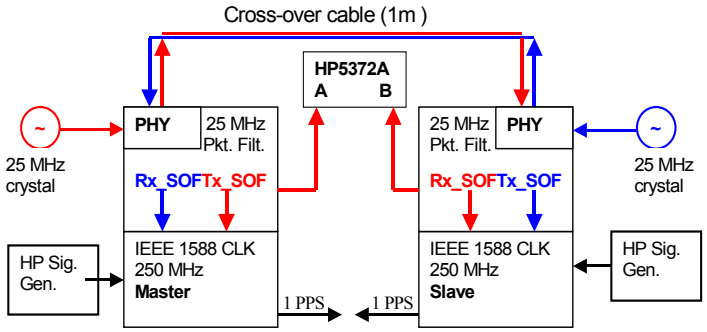


1 PPS signal

- Measurement accuracy
  - HP5372A (quantized to 200 ps)
- LAN & PHY stability
- Slew rate & noise limits (ΔV → Δt)
  - Finite rise time of 1PPS signal
  - EMI impact (Cell-phone sync.)
- Matched cable and probe lengths
- Finite duration of meas. (15 min)

Stability of LAN Transit Time

- **Do not see PHY state change issues (National Semi. DP83865BVH)**
- **Spread of Tx → Rx delay ~ 1 ns**
  - **PHY PLL bandwidth**
  - **Stability of 25 MHz PHY crystal**



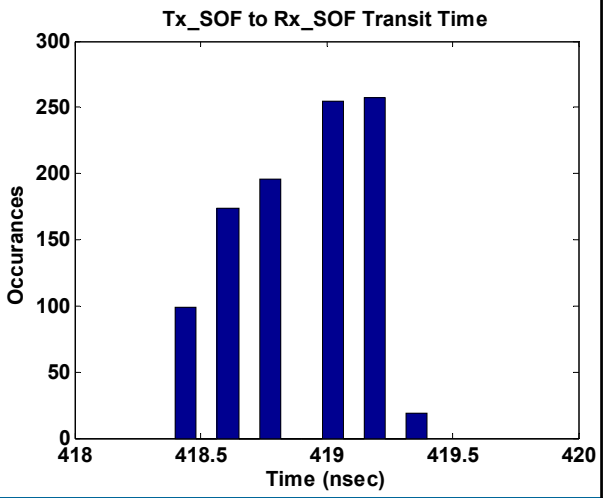
Stability of LAN Transit Time

One-way delay:

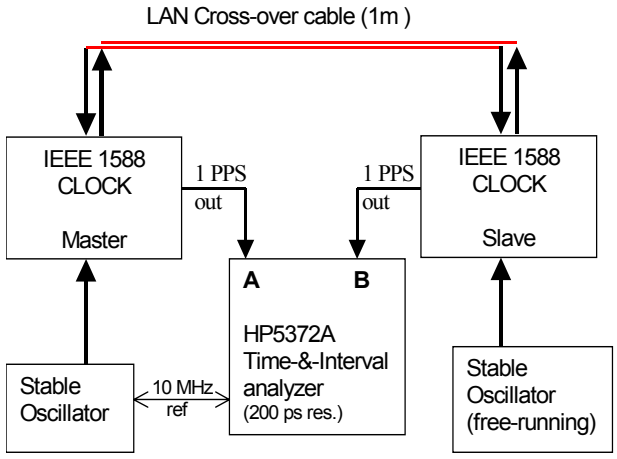
•Tx_SOF to Rx_SOF

Mean	418.9	ns
STD	0.27	ns
Max	419.4	ns
Min	418.4	ns

1000 measurements over 15 min

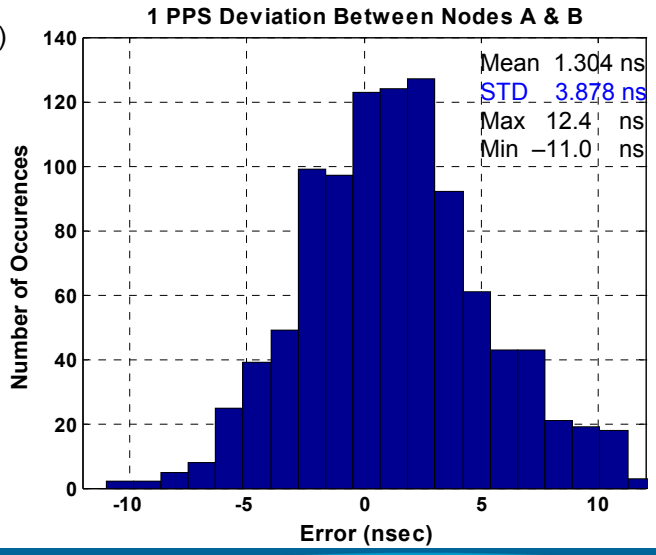


Set-up to Measure Clock Distribution



2004 Timing Results

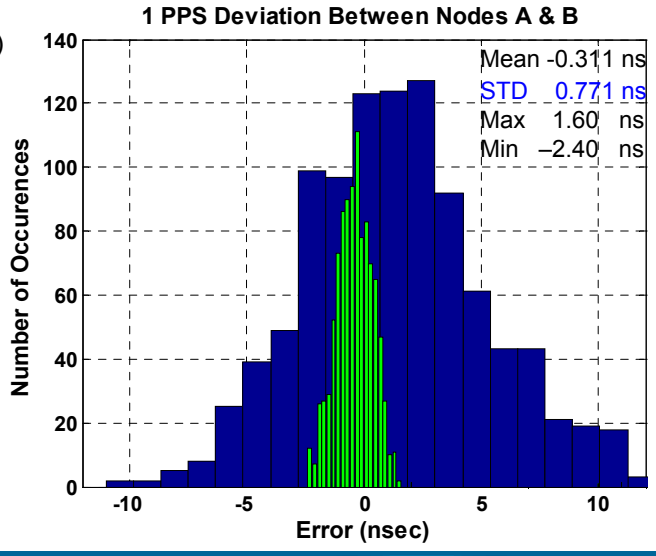
250 MHz design
(4 nsec resolution)



1000 measurements
over 15 min

2005 Timing Results

1 GHz design
(1 nsec resolution)

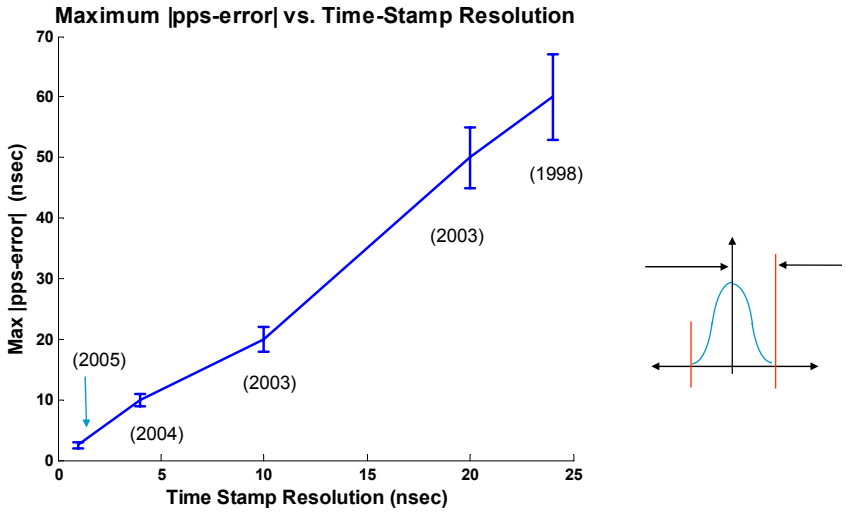


1000 measurements
over 15 min

Sources Contributing to Width of Distribution

Error Source	Max-Min error (ns)
SOF time stamp quantization	1
Pulse per second quantization	1
Rate quantization over 1sec	1
PHY jitter	1
1588 oscillators	< 0.200

Max |PPS Error| vs. Time-Stamp Resolution



Projected Limits on Time-Stamping an Edge

Time Stamp Implementation	Max – Min 5-6 σ
Programmable logic	300 ps
FPGA estimate [this work]	1000 ps
One-time programmable [Kalisz]	100 – 200 ps
Mixed Signal ASIC	1 – 5 ps
TDC: Dual DLL [Dudek, Gray]	30 , 25 ps
TDC: Multiple sine waves [Lampton]	4 ps rms
Analog trigger on scope [Agilent]	~ 1 ps

Asymmetry

Definition…

Differences in transit time of Rx and Tx LAN signals

Sources…

Standard network elements: switches, routers etc…

LAN cables

PHY (Send ← → Receive) and vendor differences

PCB routing delays (if different PCBs / vendors)

Issue…

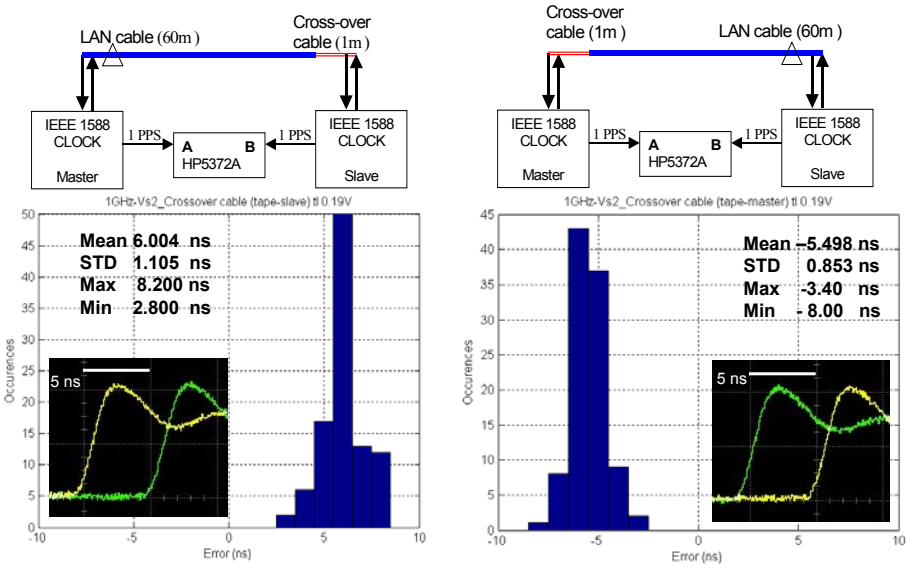
Asymmetry is limiting factor when error distribution is tight

Solution…

Boundary clocks designed for high accuracy are needed

Measure asymmetries and compensate

Impact of 60m of LAN cable



Known Systems Issues

- Distribution
  - Characteristics of oscillators
    - 1588 system oscillator
    - PHY 25 MHz oscillator and PLL
    - → Sufficient accuracy and stability at reasonable cost
- Bias
  - Measuring and removing asymmetry will become critical
    - LAN cables
    - PHY state asymmetry [Müller]
  - Calibrate out to peripherals
    - Internal latencies: 1588-clock to connectors
    - Coax cables
  - High accuracy boundary clocks
    - Availability will gate high accuracy applications

Conclusions

- Demonstrated *
  - Time-transfer with distribution below +/- 2.5 ns
    - IEEE-1588
    - 100 Base-T LAN
    - Digital FPGA solution
  - Impact of PHY jitter on one-way delay +/- 1/2 ns
    - 25 MHz crystal
    - PHY PLL bandwidth
    - Averaged by 1588 servo
  - Impact of LAN cable asymmetry
    - Solution to take out bias necessary for ns level 1588
    - No routers or switches, 1000 measurements over 15 min

Acknowledgements

John Eidson	Discussions and Guidance
Rick Baer	Discussions

References

[1] “A High-Speed Wide Dynamic Range Time-to-Digital converter”, M. **Lampton** and R. Raffanti, Rev. Sci. Instrum. Vol. 65, (11) Nov, 1994, p. 3577-3584.

[2] “A Sampling Technique and its CMOS Implementation With 1Gb/s Bandwith and 25ps Resolution”, C. Thomas **Gray**, Wentai, Liu, Wilhelmus A. M Van Noije, Thomas A. Hughes, Jr., Ralph K.I Cavin III, IEEE-Journal of Solid-State Circuits, Vol. 29 (3) March 1994, p. 340-348.

[3] “A High-Resolution CMOS Time-to-Digital Converter Utilizing a Vernier Delay Line”, IEEE Transactions on Solid-State Circuits, Vol. 35, (2) Feb. 2000, p. 240-246, Pitor **Dudek**, Stanislaw Szczepariski, John V. Hatfield, IEEE Transactions on Solid-State Circuits, Vol. 35, No.2 Feb. 2000, P. 240-247 .

[4] “Field-Programmable-Gate-Array-Based Time-to-Digital Converter with 200-ps Resolution”,Jozef **Kalisz**, Ryszard Szplet, Jerzy Pasierbinski and Andrzej Poniecki, IEEE Transactions on Instrumentation and Measurement, Vol. 46 (1) Feb. 1997, p. 51-55.

[5] “Interpolating Time Counter with 100ps Resolution on a Single FPGA Device”, Ryszard Szplet, Jozef **Kalisz**, Rafal Szymanowski, IEEE Transactions on Instrumentation and Measurement, Vol. 49 (4) Aug. 2000, p. 879-883.

[6] “PHYs and Symmetrical Propigation Delay”, Thomas **Müller**, Alexander Ockert, Hans Weibel; Proceedings of 2004 Conference on IEEE-1588 Standard for Clock Synchronization Protocol for Networked Measurement and Control, NIST-TIR 7192, p. 78.

[7] “Implementation and Performance of Time Stamping Techniques”, Hans **Weibel**, Dominic Béchaz, Proceedings of 2004 Conference on IEEE-1588 Standard for Clock Synchronization Protocol for Networked Measurement and Control, NIST-TIR 7192, p. 24.

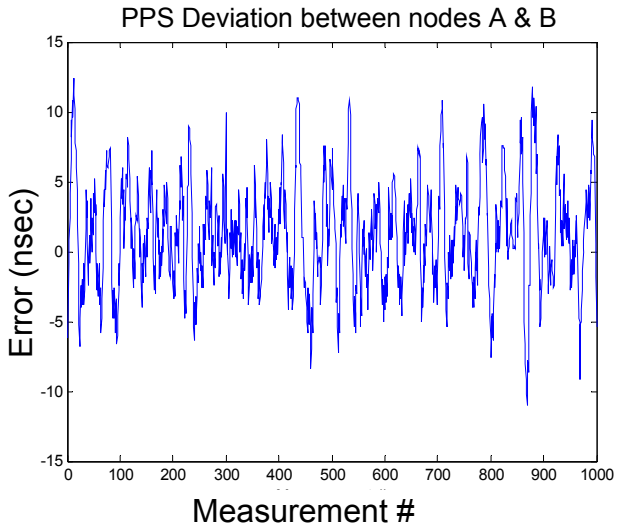
[8] Agilent **13 GHz real time oscilloscope**, DSO81304A, (40 G-samples/sec) Trace on screen to fraction of sample period which is 25ps.

See: http://www.home.agilent.com/USeng/nav/-536897734.536894885/pd.html

2004 Results

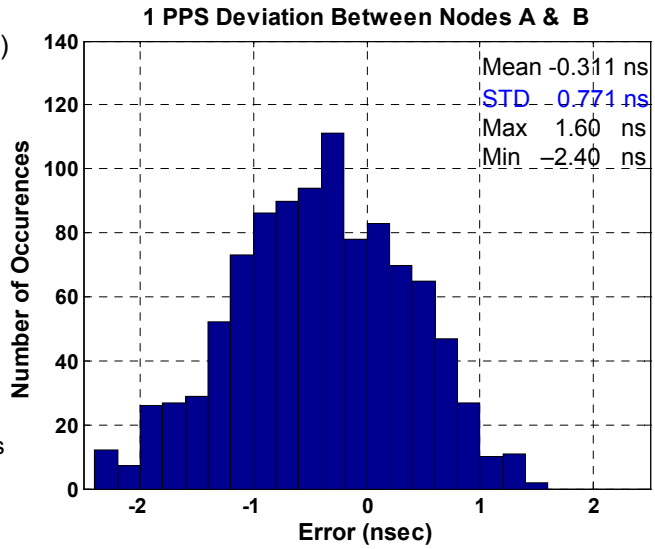
250 MHz design
(4 nsec adder)

Mean 1.304 ns
STD 3.878 ns
Max 12.4 ns
Min –11.0 ns



2005 Timing Results

1 GHz design
(1 nsec resolution)

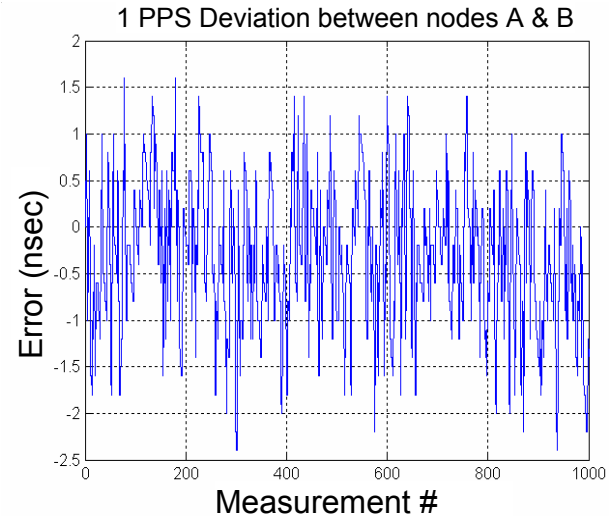


1000 measurements
over 15 min

2005 Results

1 GHz design
(1 nsec resolution)

Mean -0.311 ns
STD 0.771 ns
Max 1.60 ns
Min –2.40 ns

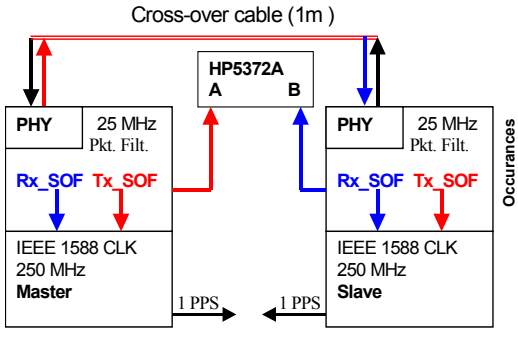


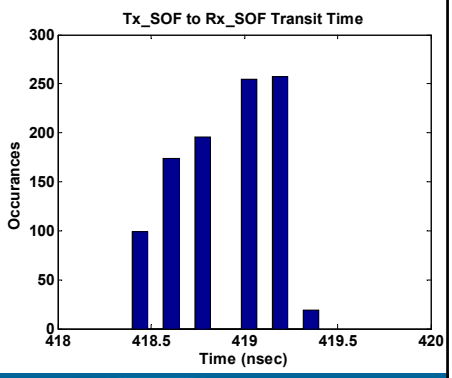
Stability of LAN Transit Time

- **Start_of_Frame signals (Master → Slave delay)**
  - **Spread of Tx → Rx delay ~ 1 nsec**
    - **PHY PLL BW**
    - **Stability of 15MHz PHY crystal**

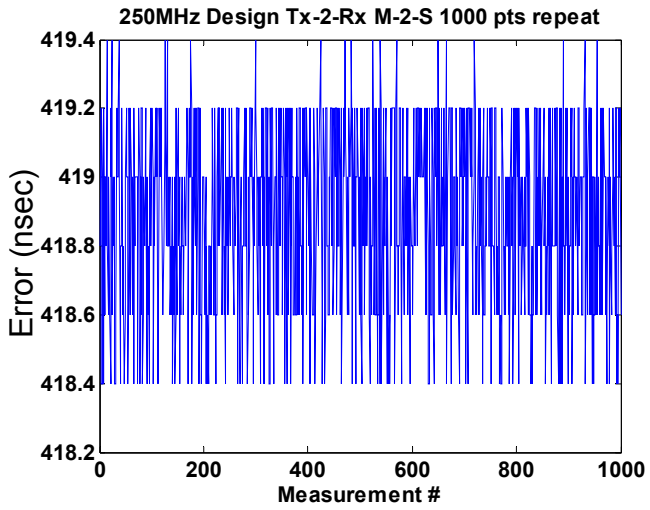
Mean	**418.9**	**ns**
STD	**0.27**	**ns**
Max	**419.4**	**ns**
Min	**418.4**	**ns**

1000 measurements over 15 min





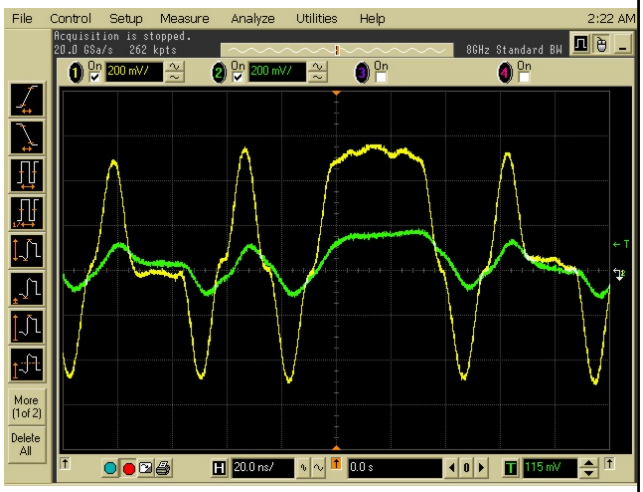
Stability of LAN Delay



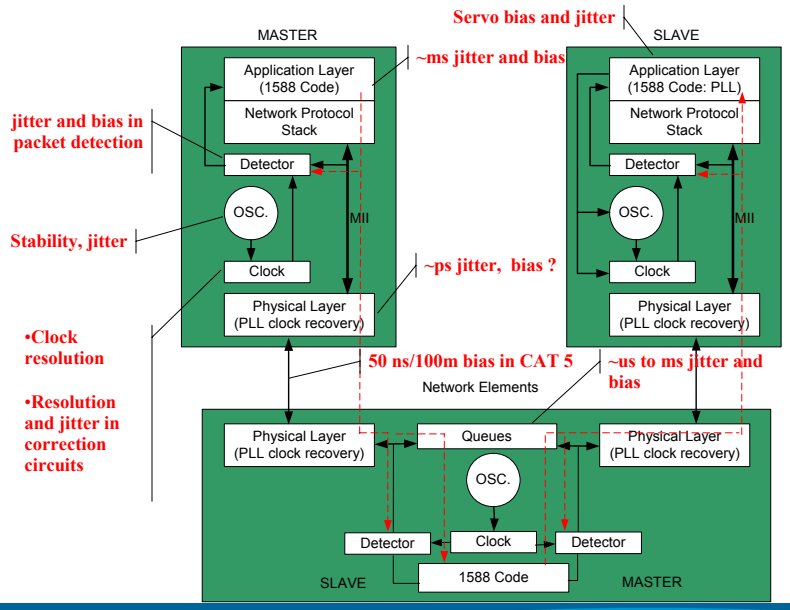
Mean	418.89 ns
STD	0.267 ns
Max	419.40 ns
Min	418.20 ns

2005 60m Cable Results

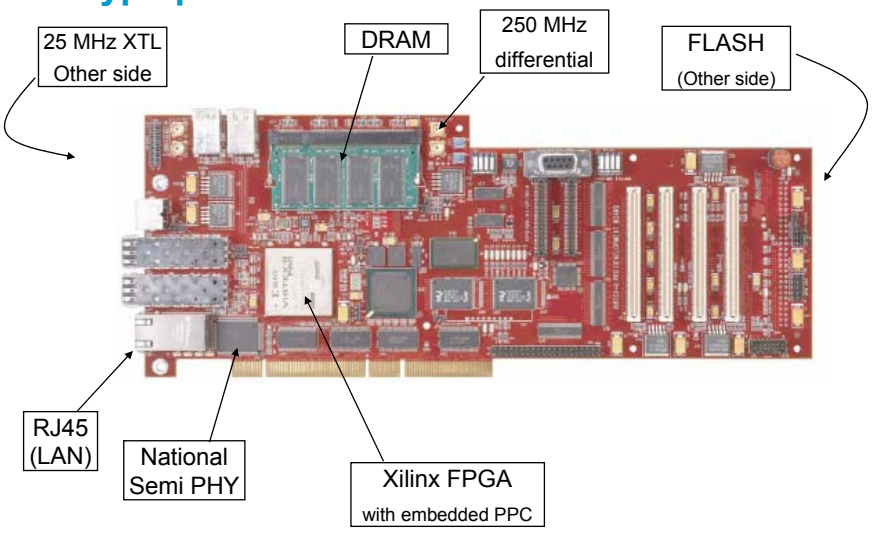
- Signal delayed 300 nsec to align
-



Background: High accuracy IEEE 1588 implementation



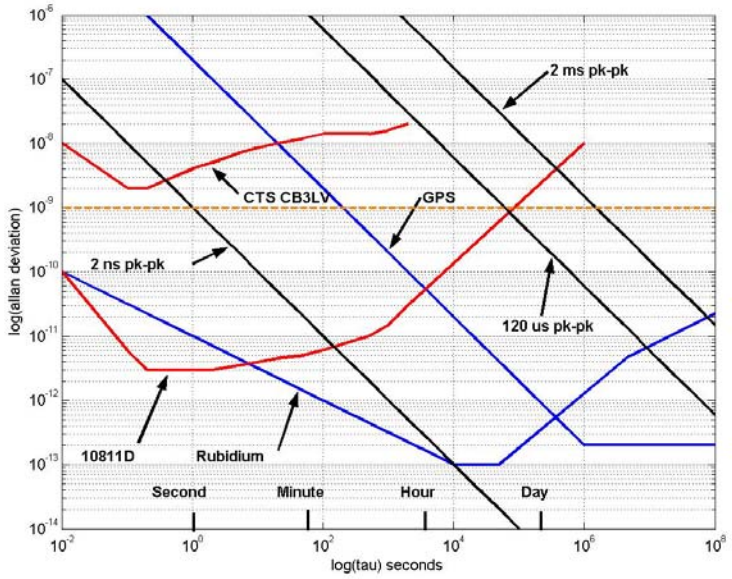
Prototype platform



Thermal considerations

- Oscillator drift is a major contributor to synchronization errors
- Quartz crystal based oscillators
  - Uncompensated oscillators generally in few ppm/degree range
  - Thermal compensation typically x10 to x100 better
- Atomic based oscillators
  - Several orders of magnitude less drift than quartz

Oscillators: Allan Deviations for Common Sources



Clock resolution considerations

- The LSB of the actual clock and any datatype representation limits synchronization accuracy
- The minimal rate and/or offset adjustment of the clock and any datatype representation limits the ability of the servo to correct to the desired accuracy

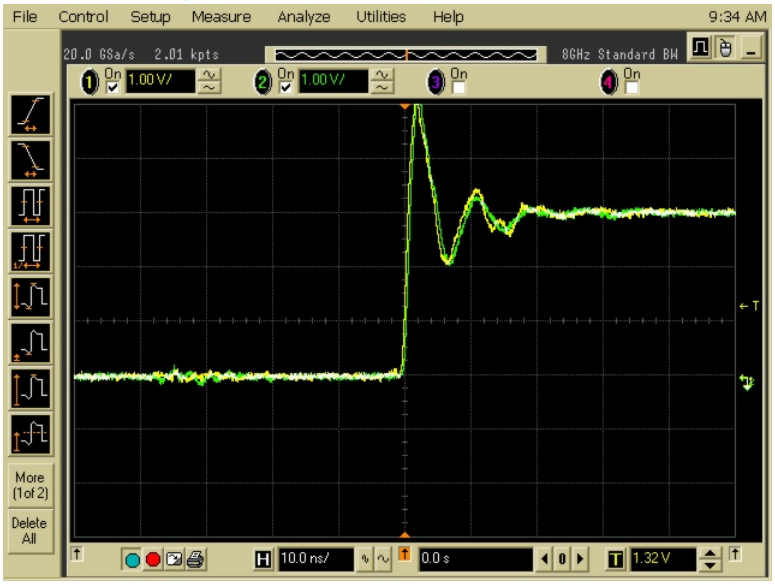
Network signaling characteristics

Signaling rates:

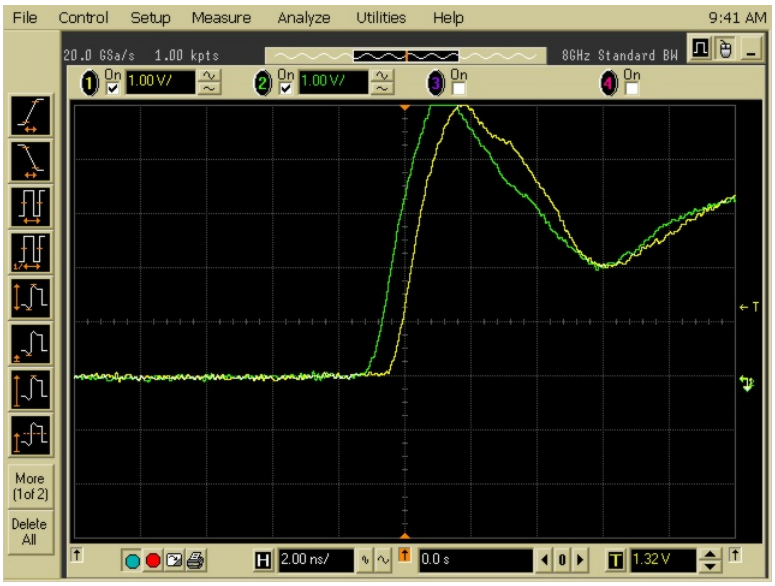
- 10 BT: 10 MHz, 100 ns period
- 100 BT: 25 MHz, 40 ns period
- 1000 BT: 125 MHz, 8 ns period

Rise time of the network signal limits the accuracy of generating a reproducible time stamp

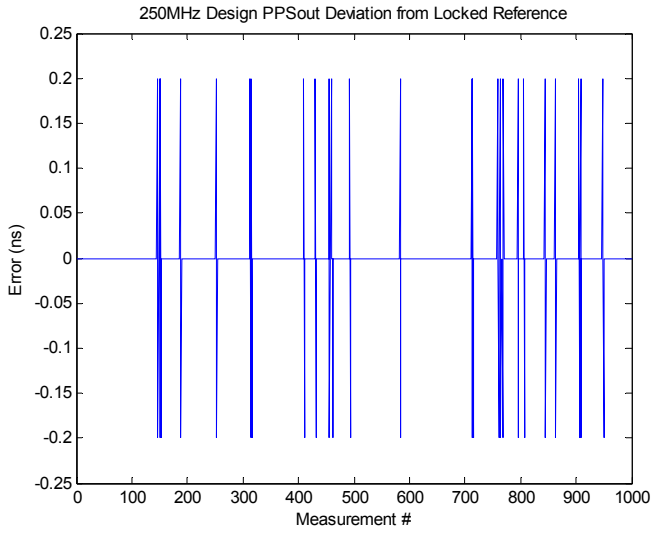
Typical PPS Sync Pulses



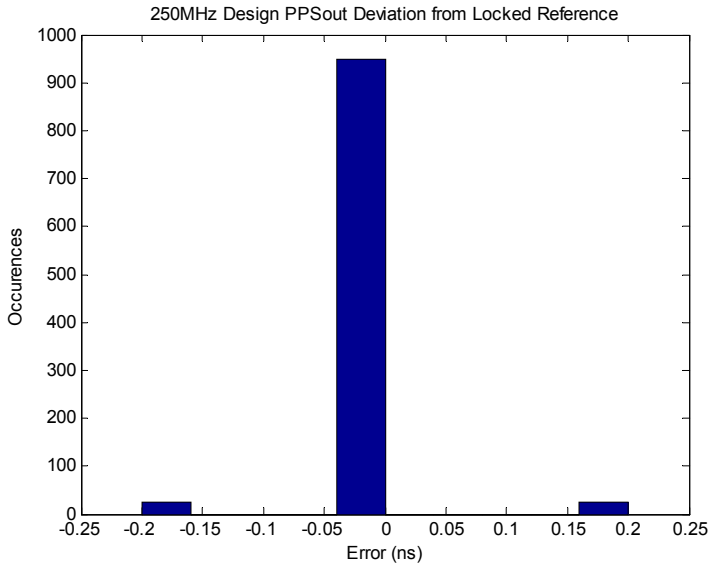
Additional PPS Screen Shot



HP 5372A Resolution



HP 5372A Resolution



Design of an FPGA-Based Hardware IEEE-1588 Implementation

John Guilford, Agilent Technologies

Abstract

The IEEE-1588 Precision Clock Synchronization Protocol has been implemented in a low cost PMC Processor board. A Xilinx FPGA sits between the CPU and the MII bus and implements the high accuracy timers and adjustable clock used for timestamping IEEE-1588 packets as well as timestamping external events and generating external trigger events. A description of this hardware is provided. Along with the software running in the PowerPC, this board provides a modular, stand-alone implementation capable of keeping time ± 20ns.

In 2004 Agilent Technologies decided to implement the IEEE-1588 Precision Clock Synchronization Protocol on an existing PMC Processor board already used in other products. This board, called the P1000, consists of an IBM PowerPC 405GP Processor

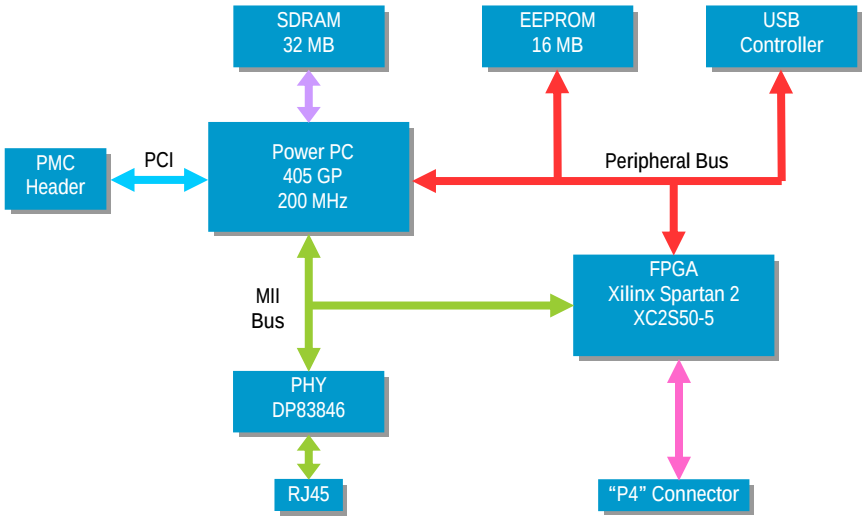


Figure 1 - P1000 Block Diagram

Figure 2 - P1000

along with memory, LAN and USB interfaces, and a Xilinx Spartan 2 FPGA (Field Programmable Gate Array). When the P1000 was designed, support for IEEE-1588 was included. The MII bus between the MAC (built into the PowerPC) and the PHY was wired to the FPGA. This allowed IEEE-1588 to be implemented without modifying the board design or layout.

Since the board was already designed, the choice of the FPGA was fixed as a Xilinx Spartan 2 XC2S50-5. This is an older device limited in speed and size. The resources available in this part include CLBs (Configurable Logic Blocks) and dual port block RAMs. A CLB consists of four 4-input LUTs (Look Up Tables), four registers, and dedicated carry logic for adders and counters. Each LUT can be configured as any logic function of four variables, a 16x1 RAM or ROM, or a 16 deep shift register. Each block RAM is a dual port memory of 4 kilobits. Each of the two ports has its own clock

and can be independently configured with different widths ranging from 1 bit by 4096 deep to 16 bits by 256 deep. This FPGA also has 176 user I/O pins, each with optional input and output registers.

FPGA Resources

- 384 CLBS (Configurable Logic Blocks)
  - 1536 LUTs, 1x16 Memories, or 16 deep shift registers
  - 1536 Registers
- 8 Dual Port Block Memories
  - Eight 4kbit ROM/RAMs
  - Each port independently sized from 1x4096 to 16x256
- 176 User I/O Pins

One complication of the design was that there are four different (essentially asynchronous) clocks used in the design.

- There is the processor peripheral bus clock, *PCLK*. This is nominally 33 MHz, and is derived from the PCI clock supplied by the host board.
- There is the main clock, *CLK,* which is used for all the timer functions. This is nominally 50 MHz and can come from two different sources. For standalone operation, the board provides a clock that is multiplied up from the PCI clock. Alternately, an external higher quality clock can be provided.
- There is the transmit PHY clock, *TxCLK*, on the MII bus. This is nominally 25 MHz and clocks the data the processor is sending to the PHY.
- There is the receive PHY clock, *RxCLK*, on the MII bus. This is nominally 25 MHz and clocks the data the processor is receiving from the PHY.

The FPGA has several tasks that it needs to accomplish. It needs to maintain an accurate representation of the current time. This includes having a steerable or adjustable timer. Since the timer's clock isn't directly adjustable (as a VCXO might be), this adjustment needs to be done digitally. The FPGA needs to monitor the MII bus in order the detect IEEE-1588 packets as near to the LAN cable as possible. When it detects an appropriate packet, it must record the current time or timestamp the packet. This must be done both for transmitted packets as well as received packets. This is sufficient for maintaining the IEEE-1588 protocol and keeping the timer in sync. To be useful, the FPGA also needs to be able to timestamp external events and to generate a precisely timed external signal or time-trigger. For diagnostic reasons, the FPGA needs to generate a 1 PPS (Pulse Per Second) signal with the leading edge of the PPS signal aligned with the second boundary. Finally, the FPGA needs to communicate with the PowerPC processor and reliably move data between the different clock domains.

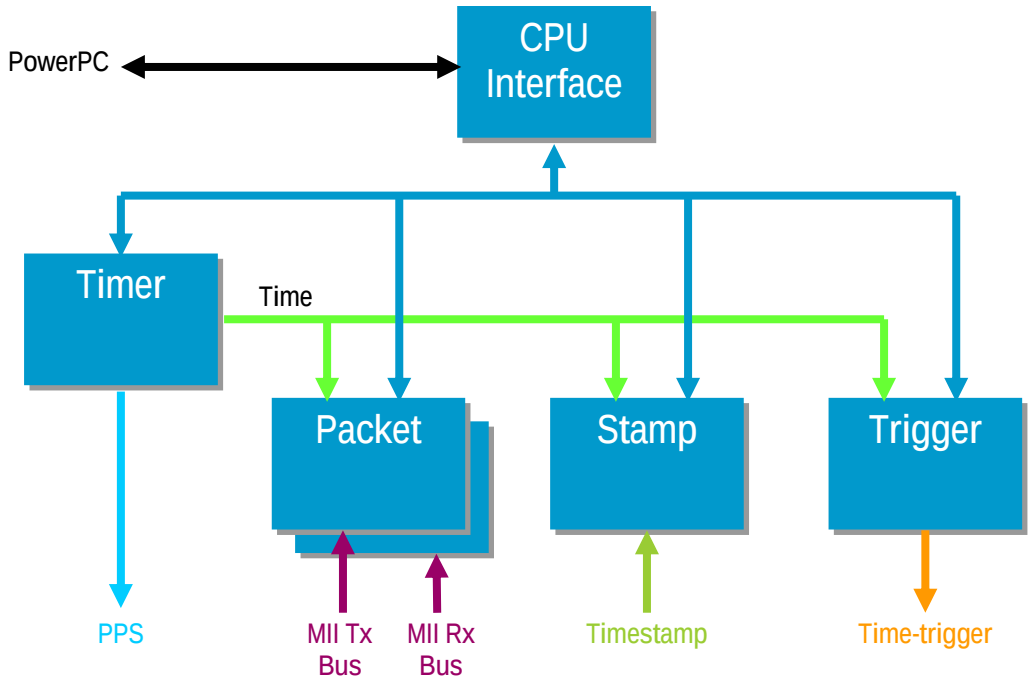


Figure 3 – FPGA Block Diagram

The design of the FPGA can be broken down into several different blocks:

- **Timer** – maintains the adjustable timer and the time of day, and generates the PPS.
- **Packet** – detects and timestamps the IEEE-1588 packets.
- **Stamp** – timestamps an external signal input.
- **Trigger** – generates an external time-trigger output.
- **CPU** – Interfaces between the PowerPC and the rest of the FPGA.

Timer Block

The timer block maintains and distributes the current time. The timer value updates nominally every 20 ns (though the exact rate depends on the exact frequency of the timer clock). The time is maintained as two 32 bit values. The first is the number of seconds since the epoch. The second is the number of *subseconds* within a second. A subsecond is close to a nanosecond and is precisely defined as 2^{-30} seconds, (approximately 1.07 ns). The choice of using subseconds instead of nanoseconds was made to make the hardware simpler (smaller and faster). Every 20ns the timer accumulates either 21 or 22 subseconds, with the long term average being 21.475 subseconds. [21.475 subseconds ≈ 20ns × 2^{30} subseconds/10^9 ns.] Though the nominal average increment is 21.475 subseconds per clock cycle, by changing the *Rate* register the average increment can be varied from 21.0 to almost 22 subseconds per clock cycle. This results in an adjustment range of ±2.2% (strictly +2.4/-2.2%). The *Rate* register is 32 bits long. A change of one unit in the *Rate* register results in a frequency change of (10^9 / 2^{32}) (10^9 / 2^{30}) / 20 = 10^{17} / 2^{63} ≈ 1/92.234 ns/s.

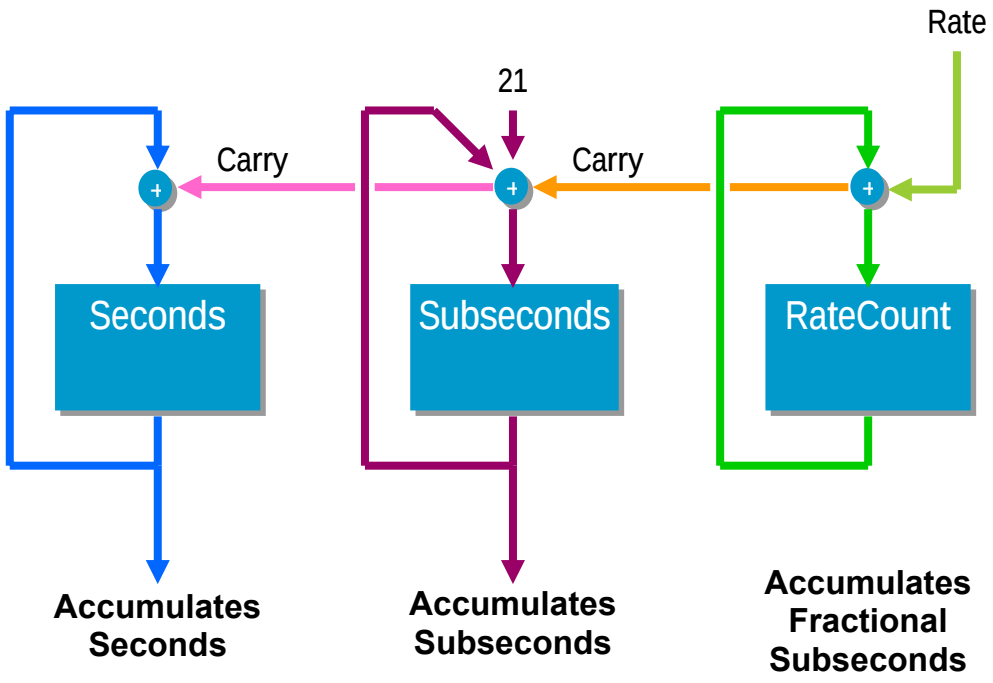


Figure 4 – Timer Block Diagram

The timer is implemented as three cascaded accumulators as shown above. The *RateCount* register accumulates fractional subseconds based on the programmed *Rate* register. If the *Rate* register is set to its smallest value, there would never be a carry out of the *RateCount* accumulator. The timer would run at its slowest rate in this case. If the *Rate* register is set to its largest value, there would be a carry out almost always resulting in the timer running at its fastest rate. Although it isn't shown in the diagram above, the carry out of the *Rate Counter* goes through a pipeline register to help the FPGA meet its timing requirements. This can be done since we're just using the average rate of the *RateCount* carry outs and not the actual value of the *RateCount* register. On every clock cycle, the *Subseconds* register accumulates the fixed value of 21 (decimal) plus the carry out of the *RateCount* accumulator. The carry out of the *Subseconds* (30 bit) register accumulates into the *Seconds* register.

Another function of the timer block is to provide a means of not only setting the time, but also precisely changing the time by a given amount (for simplicity this logic isn't shown in the above diagram). To do this, when changing the timer value, the software writes a new value for the *Seconds* and *Subseconds* registers as well as the future time (as determined by the timer output) at which these new values will be written into the timer's registers. To simplify the hardware, timer changes are only allowed on second boundaries (i.e. when the *Subseconds* register rolls over). For example, suppose the software wanted to add 2-¼ second to the timer, and that the current value of the *Seconds* register is 0x3462. The software might program the new values for the *Seconds* and *Subseconds* registers to be 0x3466 and 0x1000 0000 (note: 0x1000 0000 is $2^{30} \times ¼$). It would then program the new values to be written into the timer's registers at the end of second 0x3463. In this case the timer output would be something like:

Normally (i.e. w/out the timer update):

Seconds	Subseconds
0x3463	0x3fff ffe5
0x3463	0x3fff fffa
0x3464	0x0000 000f
0x3464	0x0000 0024

With the timer update:

Seconds	Subseconds
0x3463	0x3fff ffe5
0x3463	0x3fff fffa
0x3466	0x1000 0000
0x3466	0x1000 0015

Packet Block

The packet block's job is to monitor the MII bus between the MAC and the PHY to identify and timestamp IEEE-1588 packets. The MII bus has two nibble (4 bit) wide data paths, one for transmitted packets and one for received packets, each with its own clock. This requires two copies of the packet block, one to monitor each direction. At the start of each packet (after detecting a SOF or Start Of Frame) the current time is latched. Subsequent data nibbles are examined to decide whether this packet is an IEEE-1588 packet. If it is, the timestamp, along with enough other information to allow the software to unambiguously associate the recorded timestamp with the appropriate packet, is stored in a 64 deep circular buffer. This packet identification does not have to be perfect. If there are occasional false positives (the hardware decides that a non-1588 packet is a 1588 packet) the software can merely discard these mistakes as long as they aren't so common as to push real 1588 packet timestamps out of the circular buffer before the processor can read them.

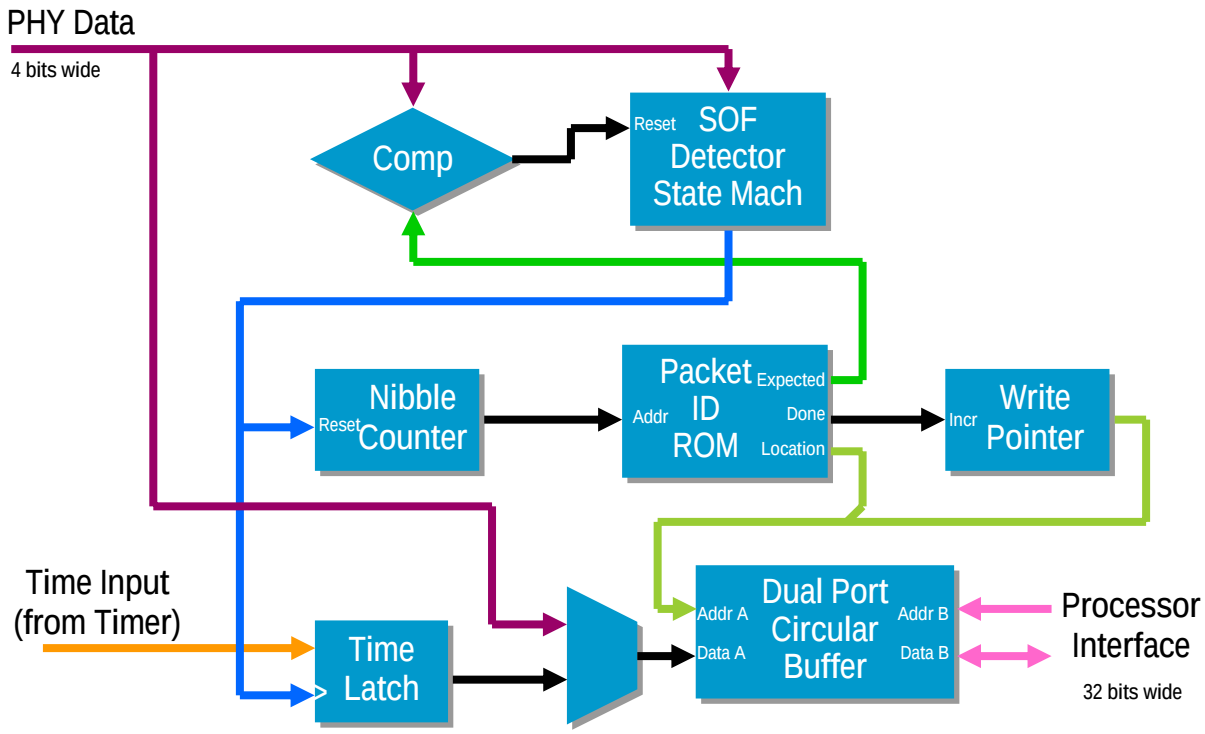


Figure 5 – Packet Block Diagram

Two other functions of the packet block are to resynchronize the data from the PHY clock and the processor clock and to deserialize the nibble wide data on the MII bus to the 32 bit wide processor data bus. Both of these functions utilize the independence of the two ports in the dual port block RAM. One port of this memory runs on the PHY clock and is configured to be 4 bits wide. This allows the packet (and timestamp) to be written into memory a nibble at a time using the particular PHY clock. The other port of the memory runs on the processor clock (*PCLK*) and is configured to be 32 bits wide (actually, since in this device the block memory only goes up to 16 bits wide, a pair of block RAMs are used to get the 32 bit width).

Between packets, the SOF (Start of Frame) detector state machine watches the PHY data looking for the packet preamble. This preamble is the sequence of nibbles consisting of a number of 1010 followed by one 1011. When the SOF is detected two things happen. One is that the current value of the Time is latched. Since this value updates on *CLK*, the SOF detected signal must be reclocked into the *CLK* domain to ensure that the Time Latch has sufficient setup and hold time to capture a valid version of the Time. Note that there is no problem copying the Time data from this latch into the dual port RAM even though these two devices are in different clock domains. By design, the output of the Time latch is static (unchanging) during the interval when it is being written into the dual port RAM. The other thing that happens after the SOF is detected is that the Nibble Counter is released from being reset and starts counting data nibbles coming in on the PHY Data bus. These nibbles correspond to the packet's header data. A subset of these nibbles need to be checked in order to determine whether this packet is a 1588 packet that needs to be timestamped and stored. If the data does need to be stored, then a different subset of nibbles needs to be written to the dual port RAM. What to do with each nibble is encoded into an entry in the Packet ID ROM. Each nibble has a 16 bit wide entry:

- Expected Value – this is the value the nibble is compared against.
- Mask – this determines whether to compare this nibble or not (not all nibbles have fixed values that need to be checked).
- Store – this determines whether this nibble needs to be saved in the dual port RAM.
- Location – this is the address offset within the dual port RAM in which to store the nibble (if it is being stored). This value provides the least significant 5 bits of the dual port RAM address while the Write Pointer provides the most significant 6 bits.
- Done – this tells the hardware that the packet is a valid 1588 packet, that the header information is stored in the dual port RAM, and that it is time to increment the Write Pointer and wait for the next packet.

If, during this process (before the hardware sees the *Done* bit asserted) the PHY Data and the expected nibble value miscompare (and the comparison isn't masked off) the hardware determines that this is not a 1588 packet and resets the SOF state machine. The hardware then waits for the next SOF. The data that has already been written to the dual port RAM, if any, is merely overwritten by the next packet's data. As

a special case, the first twelve nibble locations are reserved for storing the 48 bits of Time latch. To save storage space, only 48 out of the 64 bits of the Time value are stored, 16 bits of the seconds value and 32 bits of the subseconds value. It is up to the processor to prepend the most significant 16 bits of the seconds value. Since these 16 bits only change once every 18 hours, this is easy for the processor to do in an unambiguous manner.

Timestamp Block

The timestamp block's job is to capture the time of an external signal or event.

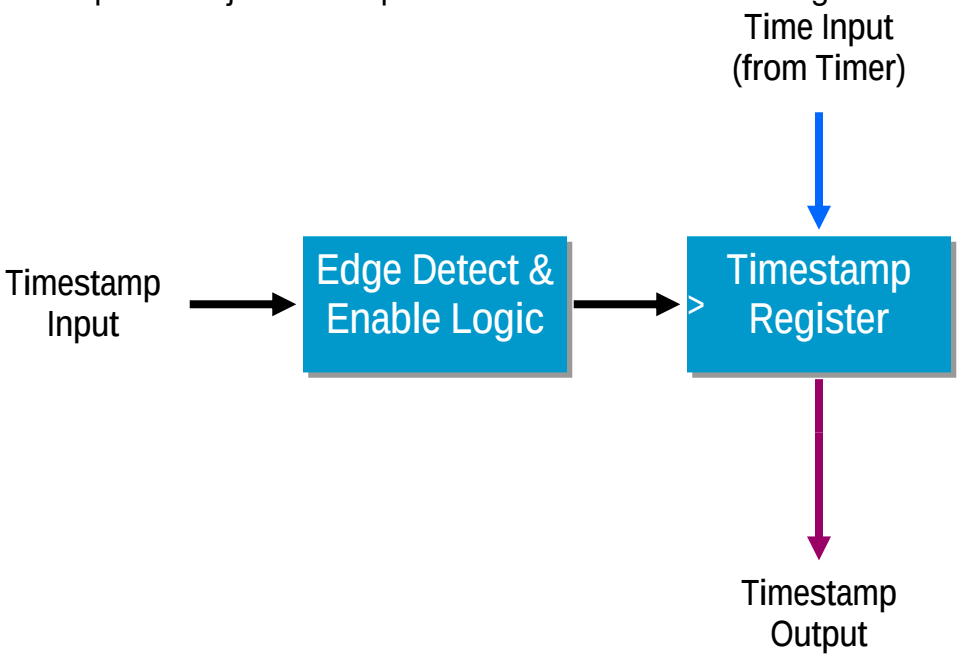


Figure 6 – Timestamp Block Diagram

This is one of the simplest blocks in the design. Once enabled, the timestamp block looks for the rising edge of the external input. Upon detection of this edge, the Time value is latched into a register. The block is then disabled until the processor re-enables it (after having read the timestamp value). The timestamp consists of:

- Time value – 48 bits consisting of 16 bits of seconds and 32 bits of subseconds.
- Sequence – 4 bits that merely count the number of external events that have been timestamped.
- Timestamp Number – 4 bits denoting timestamp number in case more than one timestamp block was implemented. In this implementation, the value is always 0000.

Trigger Block

The trigger block's job is to generate an external signal, called a time-trigger, at a programmed time.

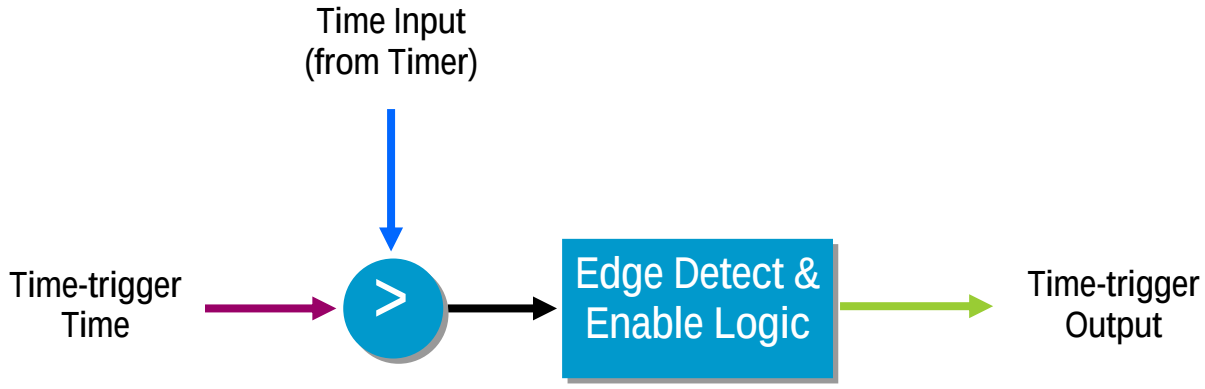


Figure 7 – Trigger Block Diagram

Since the Time value (seconds and subseconds) increment between 21 and 22 subseconds each *CLK*, not all possible values are generated. Many values are skipped over. An equality comparator can't be used as the programmed time may not be hit exactly. Instead, an inequality comparator is used and the rising edge of this comparator's output is used to generate the time-trigger output. Once the time-trigger time is programmed and the trigger block is enabled, a rising edge on the output of the time comparator will generate the time-trigger output. This output stays asserted until the trigger block is disabled by software.

CPU Block

The CPU Block's job is to interface between the PowerPC's peripheral bus and the internal registers of the FPGA. The PowerPC can be programmed to use a large variety of different bus cycles on its peripheral bus. When interfacing to this FPGA, it is programmed to use a synchronous, fixed four clock bus cycle. There are two classes of registers accessed by the CPU. Some things, such as the *Interrupt Mask* register, operate on *PCLK*. Other things, such as the *Timestamp Enable* register, operate on *CLK*. Accessing the *PCLK* registers is very straightforward. Accessing the *CLK* registers takes a bit more care due to the need to cross clock domains. When resynchronizing changing data from one clock domain to another there is the danger that some, but not all, of the bits will have propagated to the synchronizing register on any particular clock edge. In this case, the register may latch some "old" bits and some "new" bits leading to incorrect results. One solution to this problem is to assure that multiple bit data (e.g. register contents) are static or unchanging when they are resynchronized to the new clock domain. If the design only counts on one bit changing, then it doesn't matter if that change is seen on a particular clock edge or the following clock edge. There are also issues of metastability when sampling a signal that is changing near the clock edge. This is usually dealt with by using two consecutive registers to reduce the probability of metastability sufficiently.

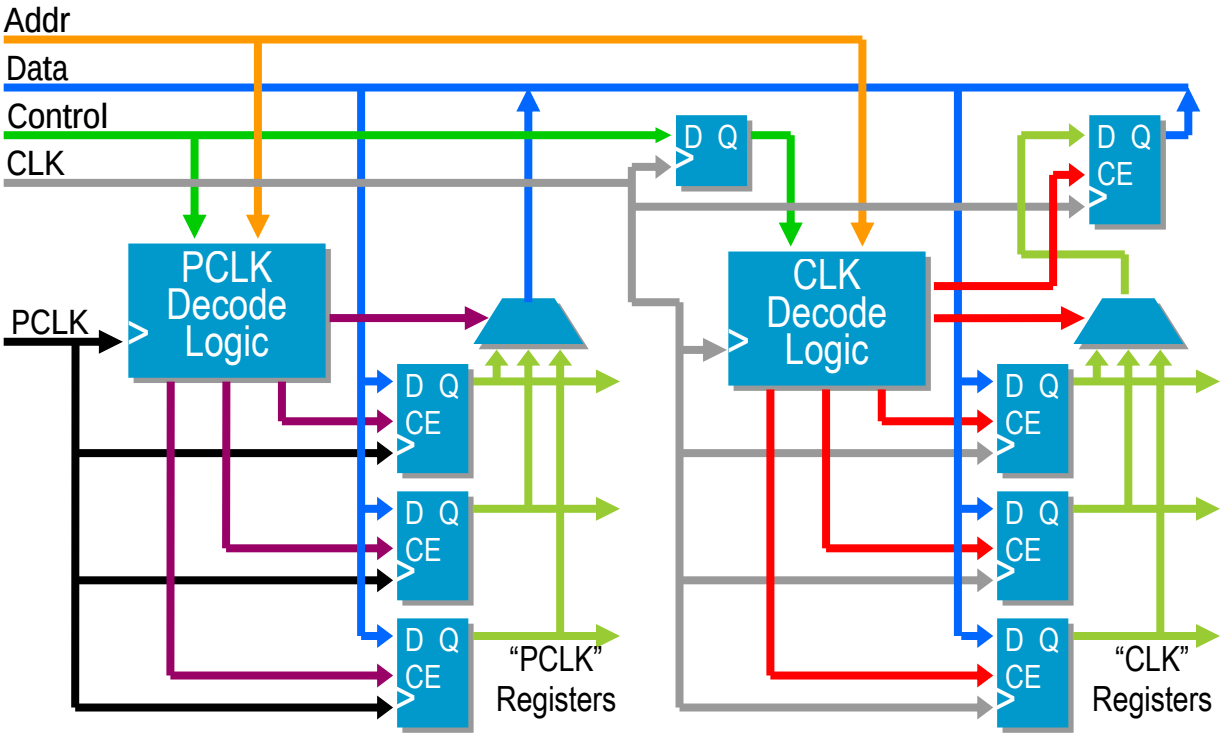


Figure 8 – CPU Block Diagram

In this design, the CS (chip select) signal is reclocked to the *CLK* domain. By the time the leading edge of the CS pulse is resynchronized, the Address and Data (for writes) have had plenty of time to propagate and settle. Write data can then be safely latched into the "*CLK*" registers. During reads from "*CLK*" registers, the contents of the register being read needs to be latched (using *CLK*) to ensure that the data being reclocked to the *PCLK* domain is unchanging when the appropriate *PCLK* edge occurs.

The CPU block has a few other tasks, such as signaling interrupts to the processor and controlling LEDs to indicate status conditions.

Example Performance

To measure the performance of this design, two P1000 modules were connected via a crossover cable. One module acted as the IEEE-1588 master and the other as the slave. The two PPS output pulses were compared to measure the relative error between the two modules.

As seen below, we observe an error distribution of about ±1 *CLK* period in width.

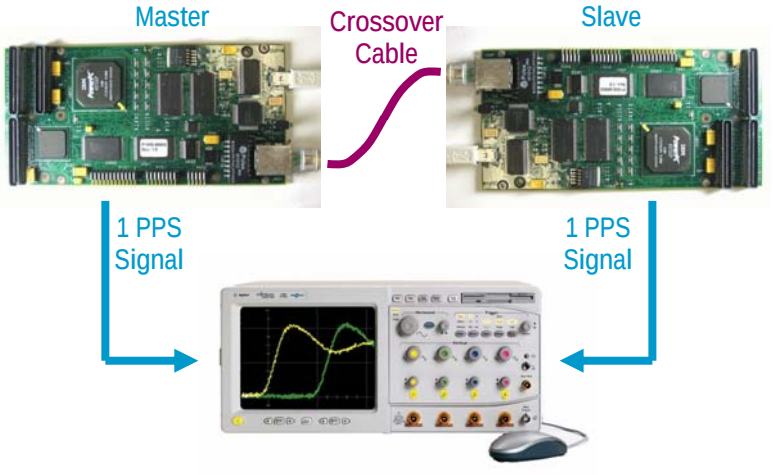


Figure 9 – Test Setup

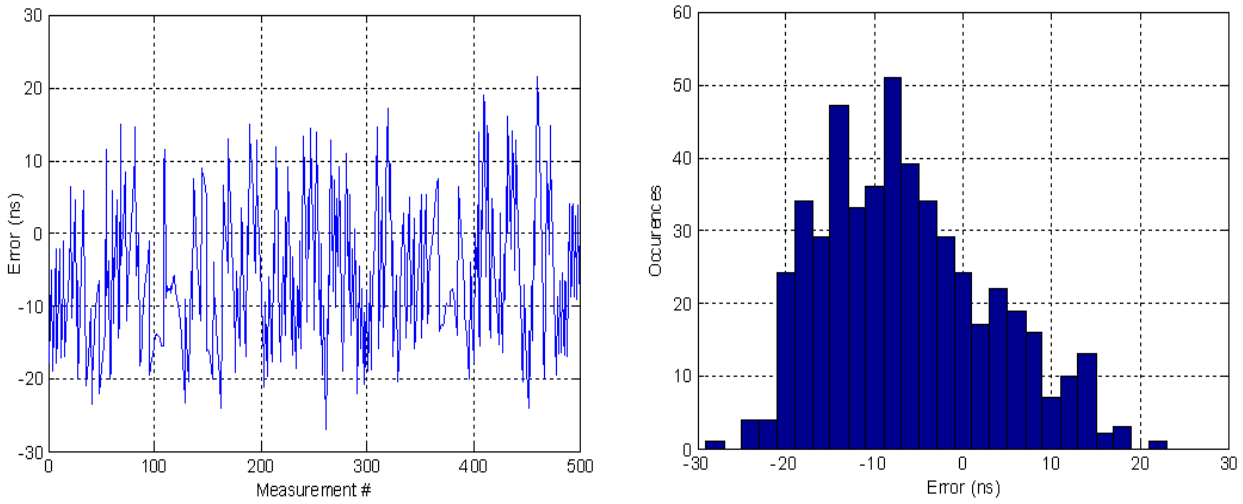


Figure 10 – Example Results

Mean (ns)	STDev (ns)	Max (ns)	Min(ns)
-6.217	9.470	21.4	-27.0

Conclusion

The IEEE-1588 Time Synchronization Protocol has been implemented in a P1000 PMC processor board. In addition to the software running in the PowerPC, a Xilinx FPGA was designed to do the precise timing functions. These functions include

- Steerable (adjustable) high resolution timer (clock)
- IEEE-1588 Packet Identification
- Timestamping LAN packets and an external input
- Time Trigger generation

This board allows one to include IEEE-1588 capability by means of a relatively inexpensive processor board, which includes a relatively inexpensive ($12) FPGA, and provides a modular, stand-alone implementation capable of keeping time ± 20ns.

In addition to being used within some Agilent internal designs, this FPGA is being provided to the LXI Consortium as a reference design to help other LXI Consortium members to more rapidly develop IEEE-1588 hardware.

Design of an FPGA-Based Hardware IEEE-1588 Implementation

IEEE-1588 Conference and Plug-Fest
Winterthur Switzerland
October 10-12, 2005

John Guilford
john_guilford@agilent.com

Measurement Research Lab
Agilent Laboratories
Everett, WA (USA)

Agilent Technologies

Outline

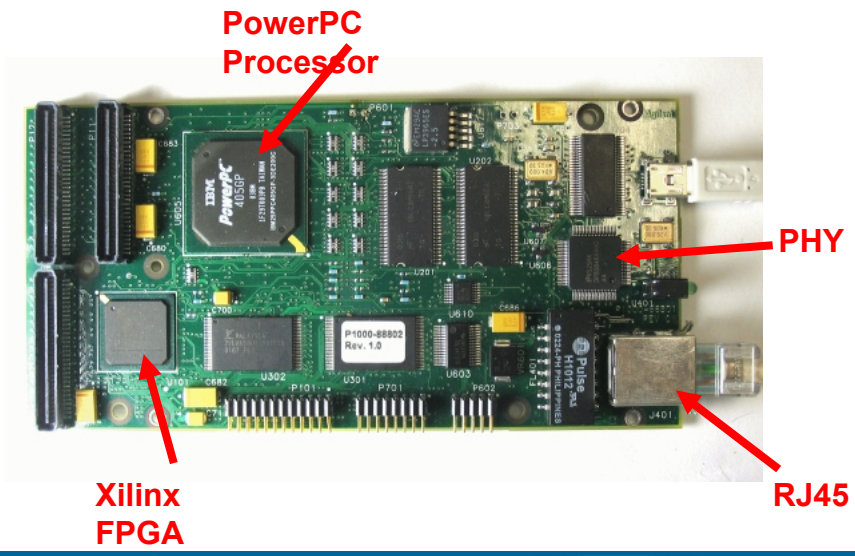
- **Project Goal**
- **Block Diagrams**
  - **Design Issues**
- **Example Performance**
- **Questions**

Project Goal

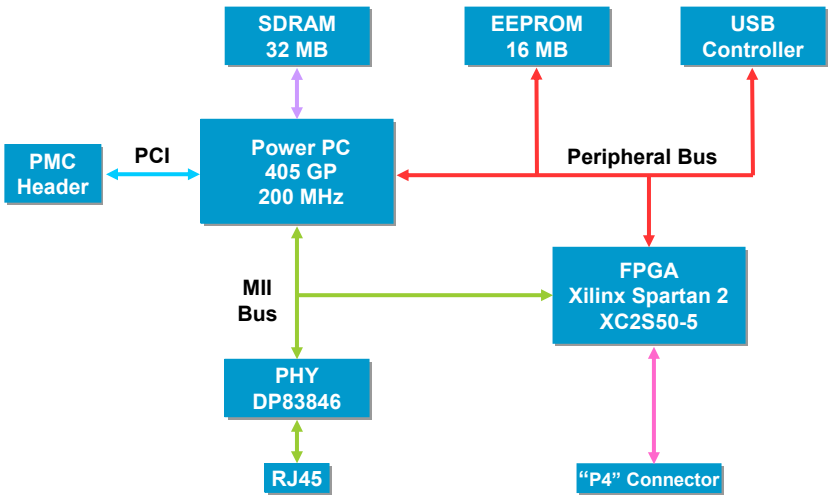
Implement a hardware assisted IEEE-1588 design using an existing PMC processor board (P1000)

- **P1000 board had been designed to allow IEEE-1588 implementation**
- **MII bus between MAC and PHY wired to an FPGA to permit the FPGA to monitor LAN packets.**

P1000 Processor Board



P1000 Block Diagram

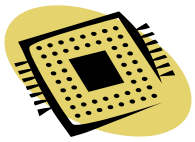


Design Constraints

- Existing hardware
- Older FPGA (Xilinx Spartan 2 family)
  - Not very large (50k gates)
  - Not very fast
- Four clock domains
  - Processor Clock (PCLK) ~ 33 MHz
  - Timer Clock (CLK) ~ 50 MHz
  - TX PHY Clock (TxCLK) ~ 25 MHz
  - RX PHY Clock (RxCLK) ~ 25 MHz

FPGA Resources – XC2S50-FG256

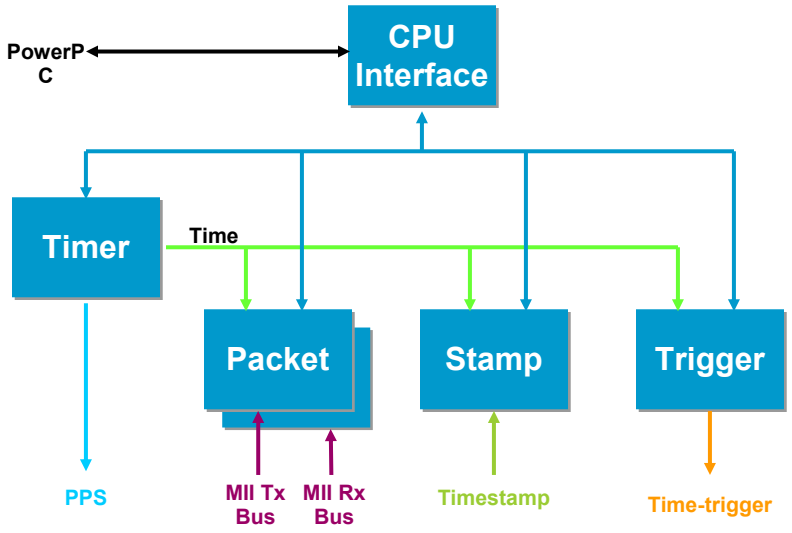
- 384 CLBs (Configurable Logic Blocks)
  - 1536 LUTs (Look Up Tables) / 1x16 Memories
  - 1536 Registers (+I/O registers)
- Dual Port Block Memory
  - Eight 4kbit ROM/RAMs
  - Each port independently sized from 1x4096 to 16x256
- 176 User I/O Pins
- Four DLLs (Delay Locked Loops)



FPGA Tasks

- Maintain a steerable (adjustable) timer
- Monitor MII bus (LAN interface)
  - Detect IEEE-1588 packets (both transmit and receive)
  - Record packet time (Timestamp)
- Timestamp external events
- Generate precisely timed external signals ("Time-triggers")
- Generate PPS output
- Interface to the PowerPC processor (CPU)

FGPA Block Diagram

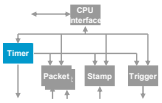


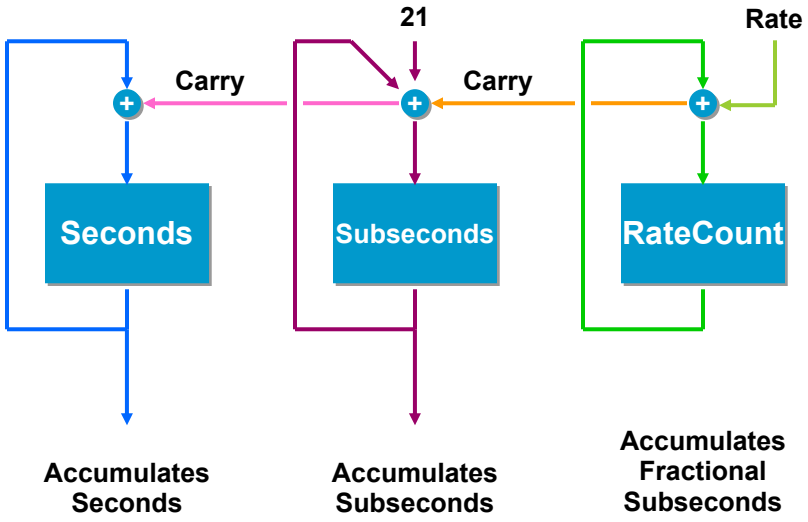
Timer Block Functionality

Generate and distribute the current time

- Timer nominally updates every 20ns
- Internal time format
  - Seconds (32 bits)
  - Subseconds (1 subsecond = 2^{-30} second ~ 1.07 ns)
- Every 20ns accumulate 21 or 22 subseconds (21.475 on average)
- Adjustment range of 0.475/21.475 or ±2.2%
- Adjustment resolution 32 bits
- Provide mechanism to set the time
- Provide mechanism to precisely change the time

Timer Block Diagram

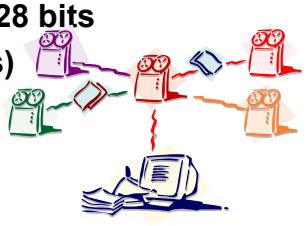




Packet Block Functionality

Two separate blocks: one each for incoming and outgoing packets

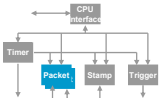
- **Monitor MII Bus (nibble wide)**
- **Detect and timestamp SOF (Start of Frame) via preamble**
- **Identify IEEE-1588 packets**
  (Identification can have some false positives as s/w can discard occasional mistakes)
- **Record data – Circular buffers 64 x 128 bits**
  - **Seconds and Subseconds (48 bits)**
  - **Packet UID (48 bits)**
  - **Port Number (16 bits)**
  - **Sequence Number (16 bits)**

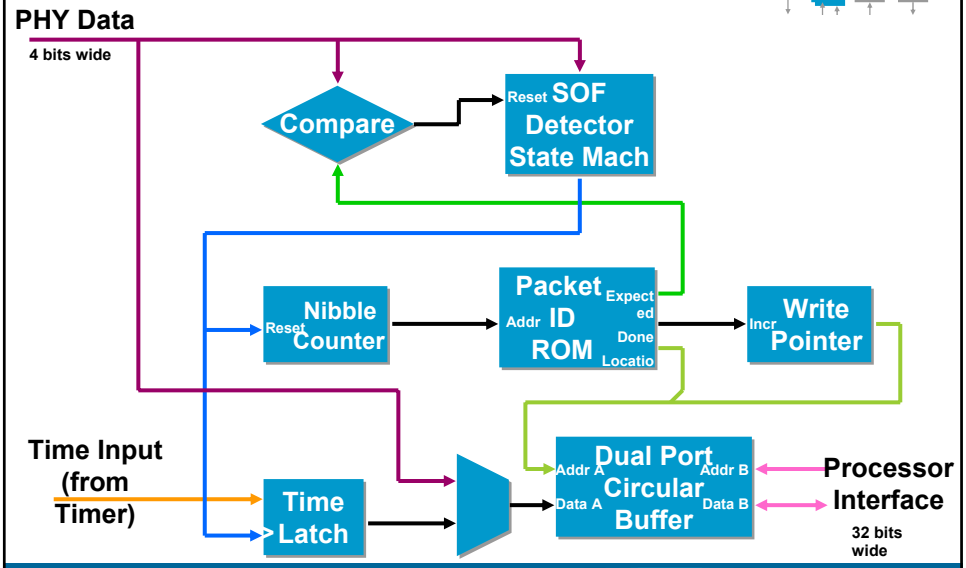


Packet Block Functionality (continued)

- Use Dual Port RAM for nibble →parallel conversion
- Use Dual Port RAM for clock domain change
  - Port A runs on the PHY Clock (TxCLK or RxCLK)
  - Port B runs on the processor clock (PCLK)
- Use Dual Port RAM as ROM for identifying 1588 packet headers
  - Defines expected nibble data values
  - Defines what data to store in circular b[illegible]er

Packet Block Diagram





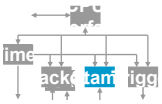
Timestamp Block Functionality

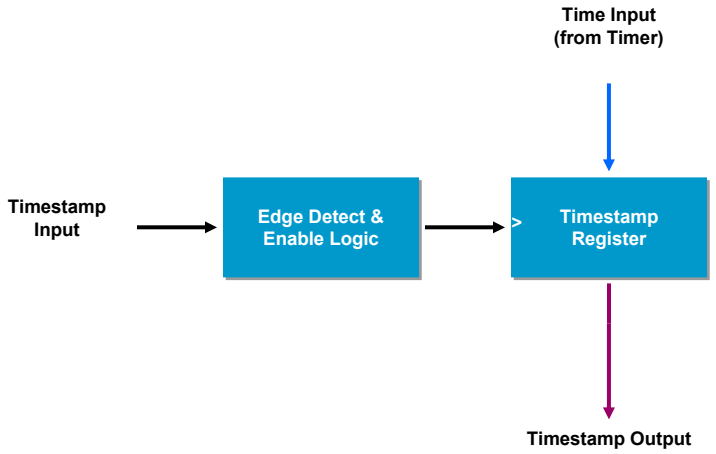
Capture the time of (Timestamp) an external signal

- Timestamp is 64 bits
- Internal time format
  - Sequence Number (4 bits) – increments w/each timestamp
  - Seconds (16 bits) – processor needs to prepend 16 MSBs

    (MS 16 bits of seconds only change every 18 hours)
  - Subseconds (32 bits)
- Only one timestamp captured each time timestamp enabled

Timestamp Block Diagram



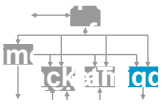


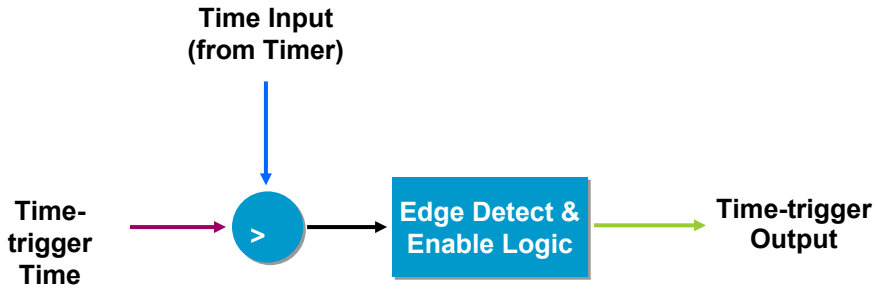
Trigger Block Functionality

Generate an external signal (Time-trigger) at a set time

- **Note: timer may skip over programmed time**
- **You can't count on the time agreeing exactly**
- **Time-trigger stays asserted until cleared by software**

Trigger Block Diagram



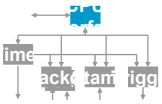


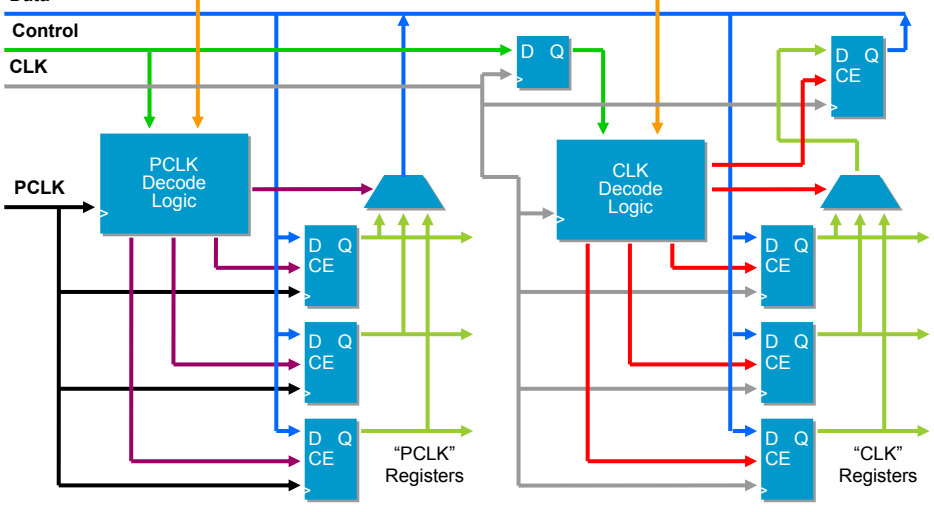
CPU Block Functionality

Interface to the PowerPC's peripheral bus

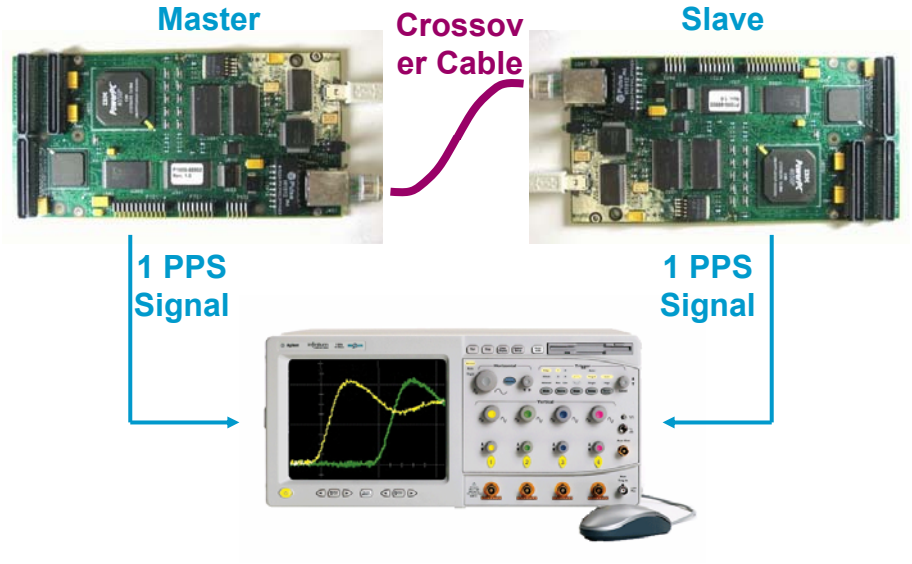
- **32 bit data bus**
- **Synchronous four clock bus cycle**
- **Maintain registers for controlling the hardware**
  - **Some registers are synchronous with processor clock**
  - **Other registers are synchronous with timer clock**
- **Resynchronize data transfers across clock domains**
- **Maintain status and interrupt registers**

CPU Interface Block Diagram

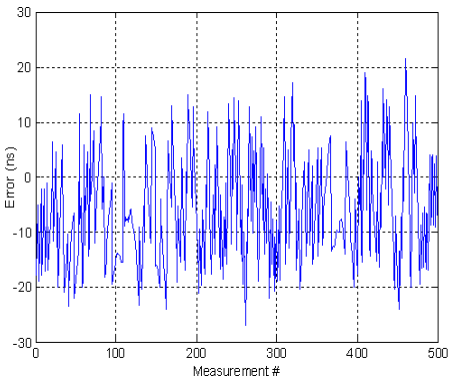


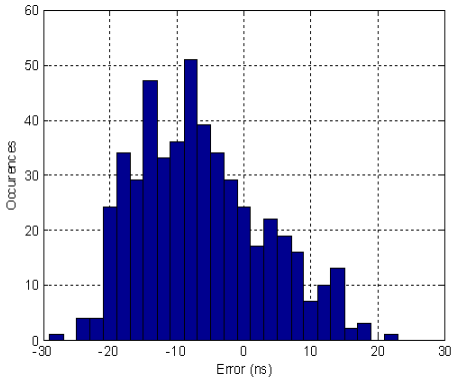


Example Performance – Test Setup



Example Performance – Results





Mean (ns)	STDev (ns)	Max (ns)	Min(ns)
-6.217	9.470	21.4	-27.0

Design Used In

- Internal Agilent Designs
- LXI Consortium Reference Design

Questions?

AUSTRIAN ACADEMY OF SCIENCES

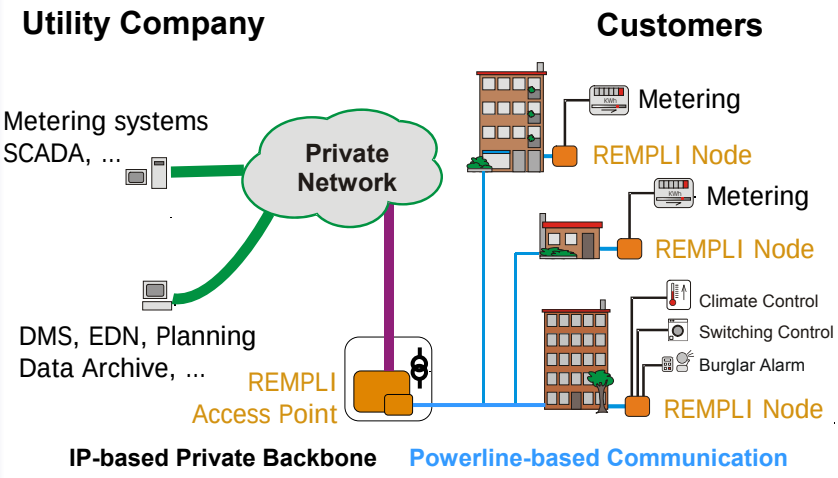
IEEE1588 Clock Synchronization in non-Ethernet Networks

Georg Gaderer,
Hannes Muhr,
Thilo Sauter, and
Nikolaus Kerö

Course of Talk

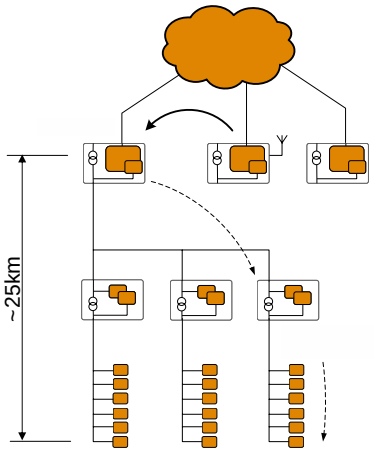
- Introduction
  - Historical Issues
  - REMPLI Project
- Requirements and problem definition
  - Network Topology
- Implementation
  - Synchronization Concept
  - Syn1588 Core
  - IEEE1588 for Powerline
- Preliminary Results
- Conclusion

REMPLI System Overview



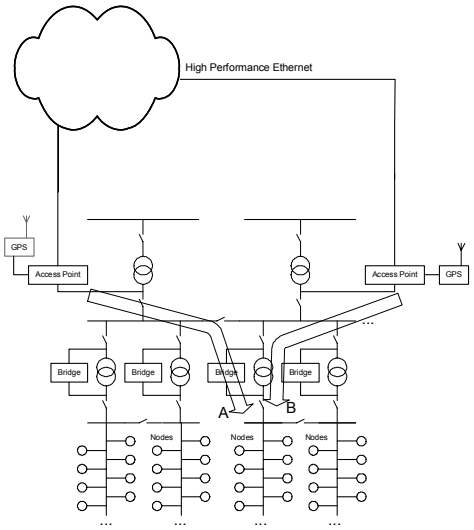
Clock Synchronization Concept- Requirements

- Clock Synchronization is needed for PLC and Application
  - TDMA, Re-Logon
  - Fraud Detection, Metering
- Intranet path:
  - IEEE1588 to synchronize APs
- PLC part: Master/Slave style between APs, bridges, repeaters and nodes
  - Transient topology requires fast logon-logoff
  - Asymmetric delays require a modification of existing algorithms



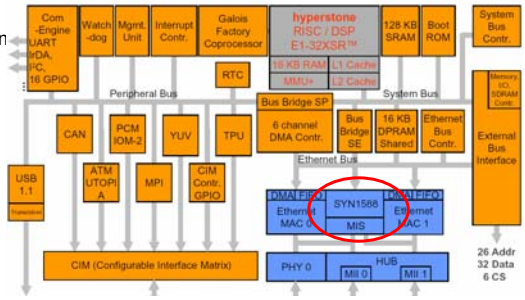
Network topology

- Non-stable topology
  - Suppliers switch subnets for load balancing
  - Nodes may be reachable (synchronized) by multiple APs (with multiple accuracies)
  - Nodes, subnets may be switched from one domain to another



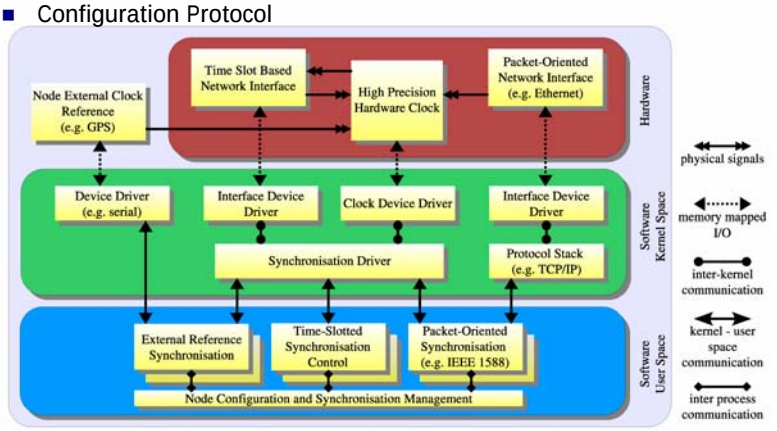
Syn1588 Core

- Implemented as IP-core in HyperStone Hynet32{X|I|S} processor
  - Commercially available in December
  - On-Chip Ethernet Phy
  - UMC 0,18µm Technology
  - TBGA 256 Case
  - 1,8V Core Voltage
  - Syn1588 core
    - 60500 Gates
    - Accessible via Ethernet Bus
    - Free configurable MII-Scann
      - Port
      - Packet Type etc.
- uCLinux 2.4
  - Compliant Device Driver



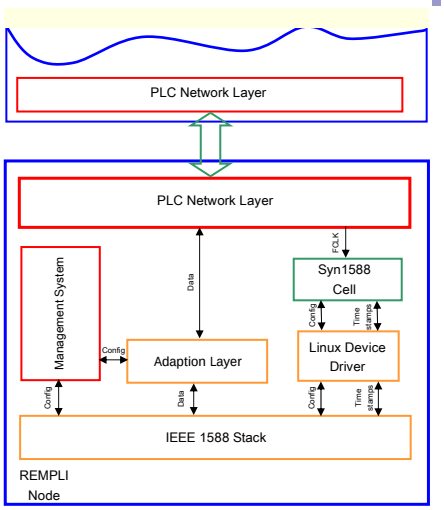
Synchronization Driver

- Synchronization Device Driver
- Synchronization Driver
- TCP/IP Protocol Stack
- Clock Synchronization stack
- Configuration Protocol



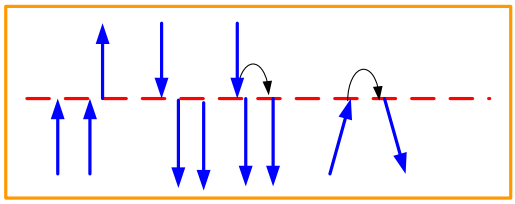
IEEE1588 for PLC Side

- **Drawbacks of IEEE1588 in PLC**,
  - Limited transmission bandwidth
  - Poor reliability
  - Non-stable topology
  - Asymmetric Delay
  - TDMA oriented network
- But with...
  - PLC-receiver side generated follow-up packets
  - Fixed-Value Delay-Response Packets
  - PTP is possible!
- Advantage:
  - Full IEEE1588 stack is usable
  - Slave may act as fallback-Master

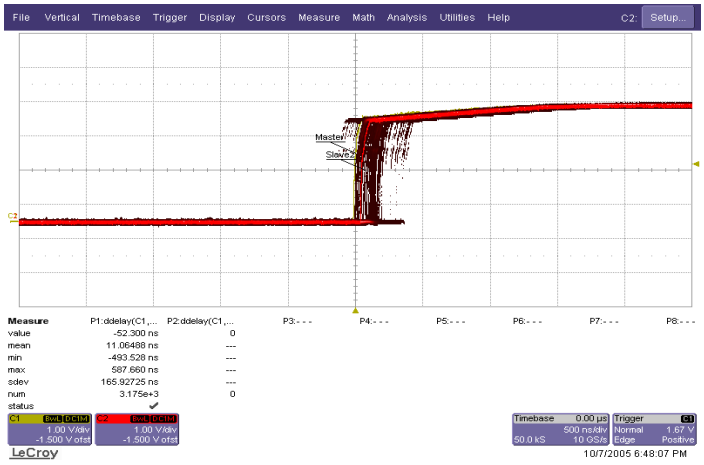


Adaptation Layer Details

- Clock synchronization (and Applications) have to deal with PLC
  - Typical approach: Translation on layer 3 and above
  - Examples
    - Security
    - Metering, for non commercial use

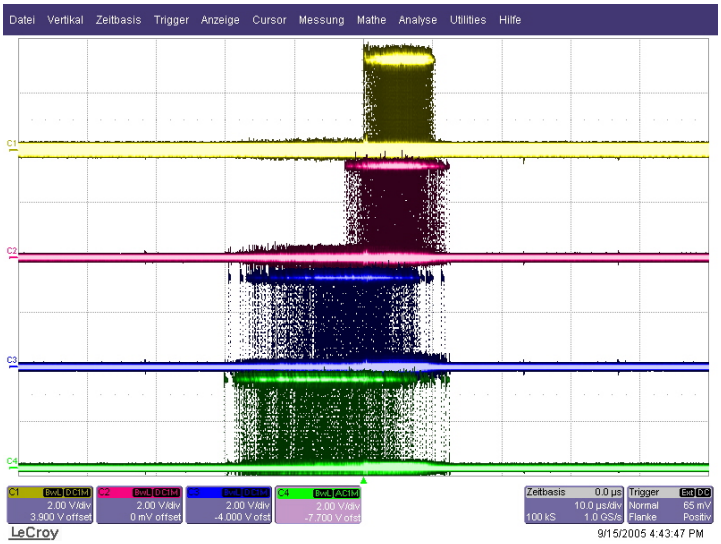


Preliminary Result: Ethernet-Part Synchronization



Note: No Delta-TS, 10MBit, 1 Master, 1 Slave

Preliminary Result: Clock-Jitter with multiple Repeaters



Conclusion

- Powerline has many disadvantages when using IEEE1588
  - Non-Stable topology
  - Poor Bandwidth
  - Non-symmetric delay
- IEEE1588 is possible for PLC
  - Adaptation Layer
    - Delay-Request will be answered with fixed values
    - Follow-Up packets are generated by driver
  - Synchronization better than 880µs of small cities, with existing cabling infrastructure
- HyNet32XS with syn1588 cell
  - IEEE1588 Format-Timestamps
  - Driver Infrastructure

Chip-Design Building Blocks for Precision Clock Synchronization in Ethernet Networks

Roland Höller, Georg Gaderer, Hannes Muhr
Nikolaus Kerö

ICT
Institute of
Computer Technology

Requirements for 10ns Synchronization

- Hardware assisted timestamping
- 100 MBit/s switched Ethernet
- Extremely low clock granularity
  - High (internal) frequency for local clock
- High clock resolution
- Stable local oscillator
  - TCXO, MCXO, OCXO
- “Reasonable” sync interval length
- Residential time jitter on switch
  - Transparent Clock

System Design Constraints

- Purely digital design
  - Only PLLs as analog building blocks
  - Vendor independent design style
- Synthesizable for likewise for ASICs and FPGAs
- Existing Hardware platform as a starting point
  - ALTERA Stratix II FPGAs
  - 4-port Switch
- Independent of network load

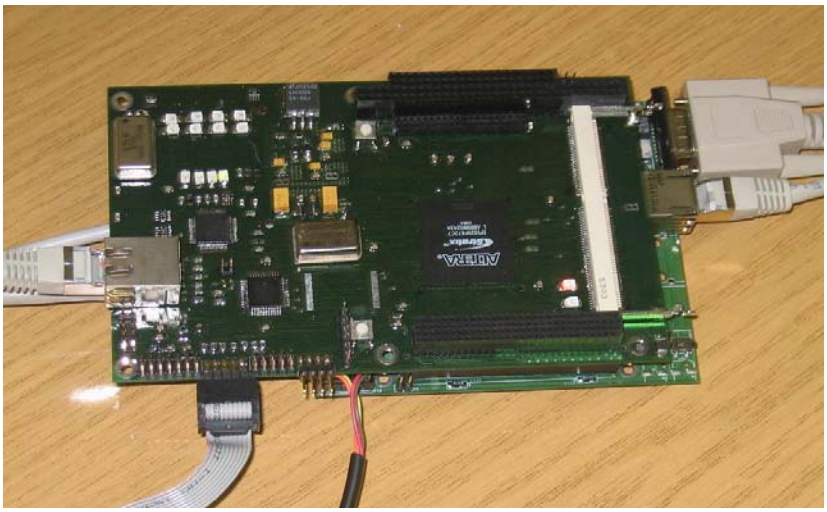
Clock Synchronization Constraints

- How to achieve 10 ns accuracy ?
- Overall accuracy π depends on

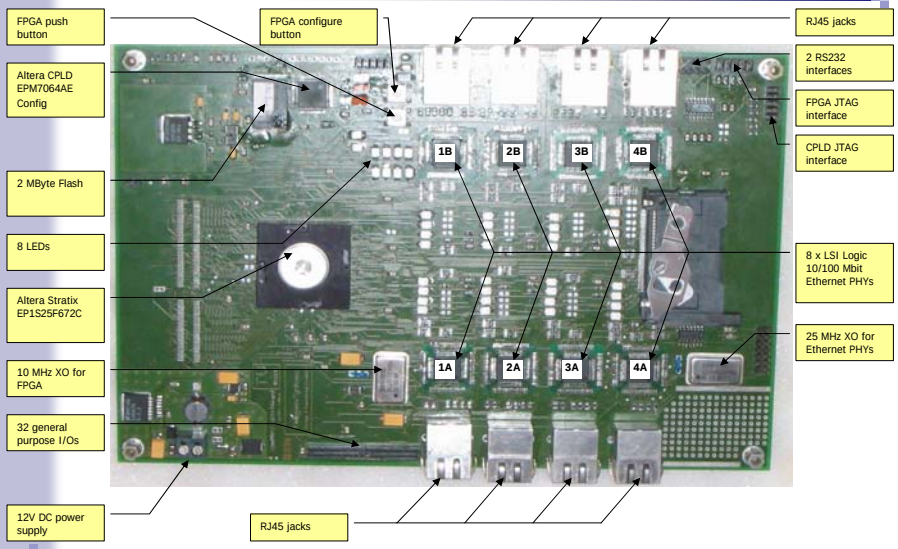
$$\pi = c_1\varepsilon + c_2 G + c_3 u + c_4 P\rho$$

- ε ... Transmission delay uncertainty
- G ... Local clock granularity
- u ... Rate synchronization uncertainty
  - Timing error due to discrete rate adjustment ($u=1/f_{osc}$)
- Pρ ... Clock drift during a re-synchronization period
  - P ... length of the re-synchronization period
  - ρ ... oscillator drift

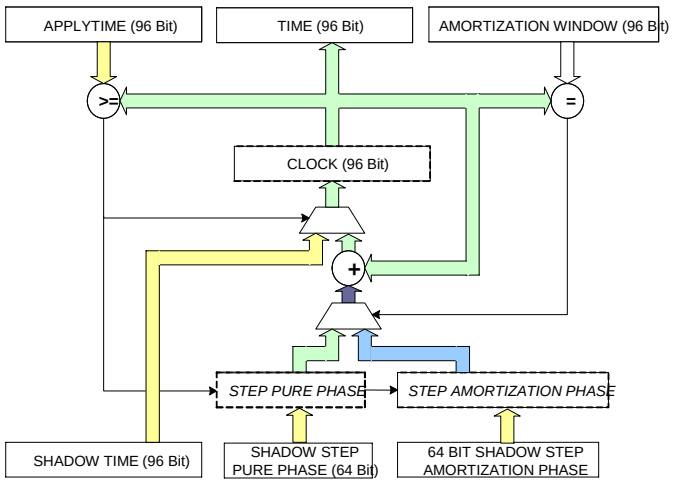
SynUTC Network Interface Card



IEEE1588/*SynUTC* Transparent Switch

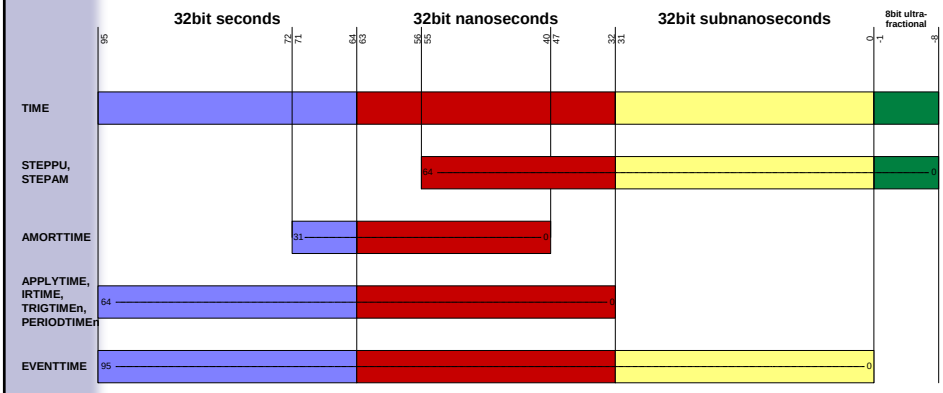


SynUTC Adder Based Clock Structure



SynUTC Adder Based Clock 2

IEEE 1588 + PSynUTC Adder Based Clock Register Layout



System Design Considerations

- Hardware Timestamping
  - IEEE1588 units reside within MII
- Local Oscillator Drift
  - 10 MHz OCXO was used
  - Clock rate sync algorithms were used
- Clock Resolution
  - 96 bit internal time representation
- Clock granularity
  - Adder Based clock running with 100 MHz
    - Limited by FPGA technology available for the design

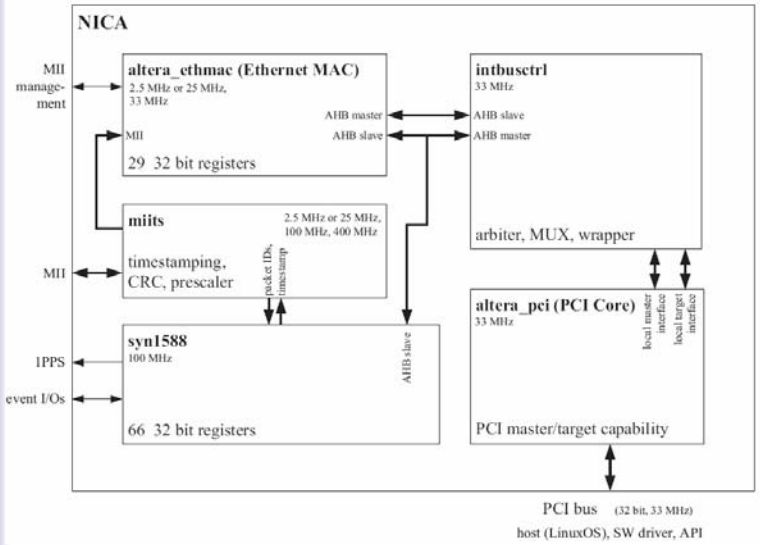
Time Stamp Accuracy

- Time Stamping at MII
  - Time Sync unit between Physical layer IC (PHY) and Media Access Controller (MAC)
  - Transmit frequency 25 MHz @ MII
- Different clock domains
  - PHY has internal PLL which syncs on data rate of sender
  - IEEE 1588 unit uses high accuracy clock domain
  - Clock domain transition introduces inaccuracies
- High frequency sampling

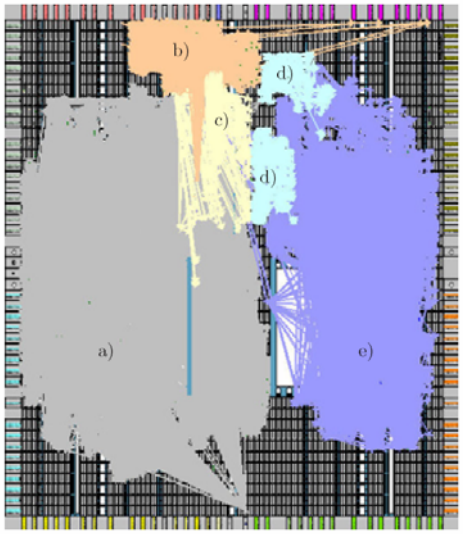
IP Core Functional Blocks

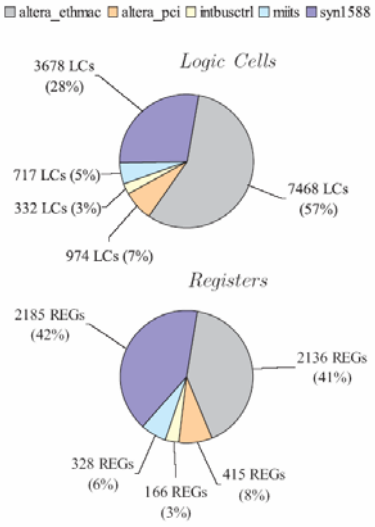
- Time Stamp Unit
  - MII Interface, FIFOs
- Syn1588 Adder Based Clock
  - Configuration and control registers
    - Increment, accuracy, amortization timers, ...
  - (Period) Timers
  - External event timestamp registers
- Internal Bus Interface Controller
  - AHB, AMBA, WISHBONE, ...

Syn1588 Aware Network Controller

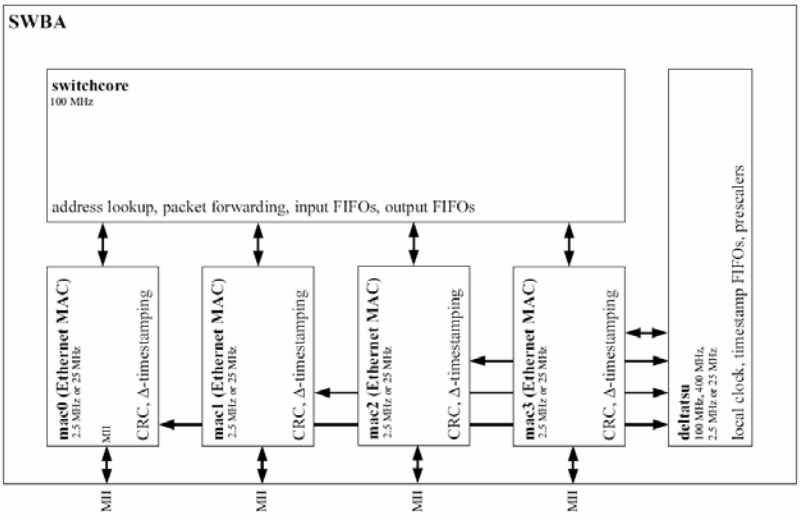


FPGA Implementation Result

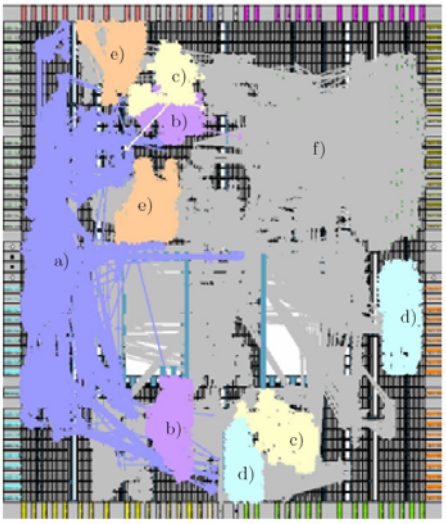


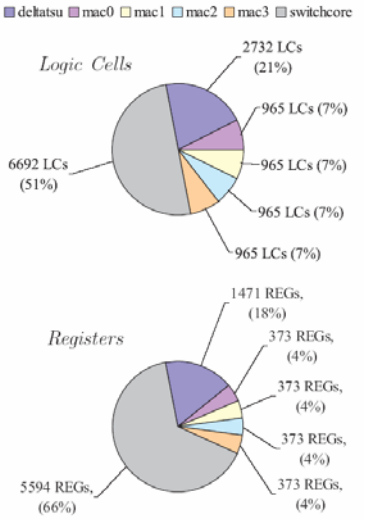


Transparent Clock Block Structure



FPGA Implementation Result

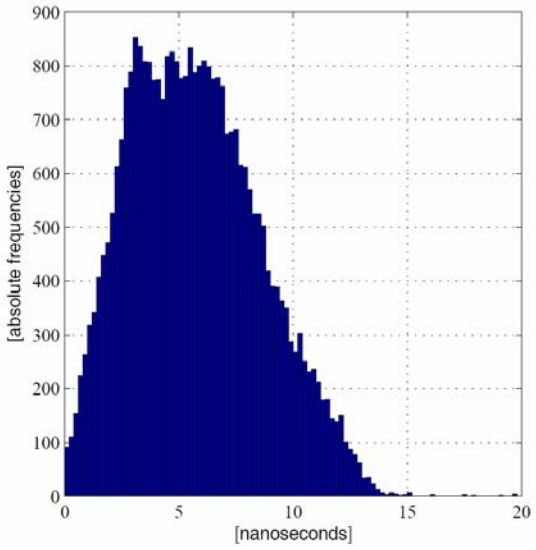




Implementation Statistics

- Network node
  - MAC: 7468 LCs 2136 REGs
  - PCI: 974 LCs 415 REGs
  - MIITS: 717 LCs 166 REGs
  - Syn1588: 3678 LCs 328 REGs
- Transparent Switch
  - DeltaTSU: 2732 LCs 1471 REGs
  - 1x MAC: 965 LCs 373 REGs
  - SW-Core: 6692 LCs 5594 REGs

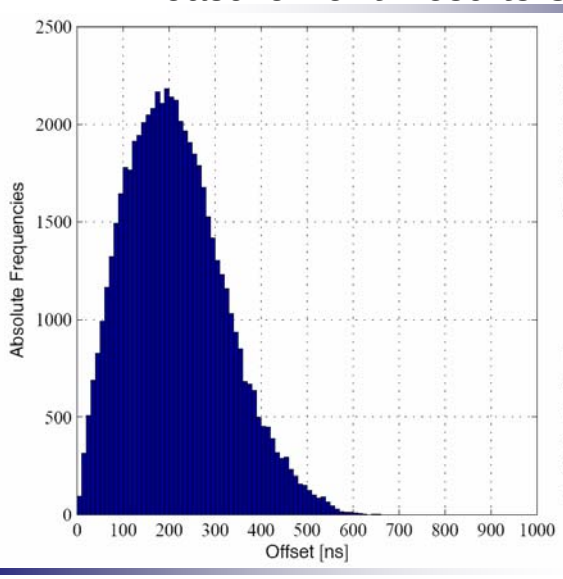
Measurement Results Hardware TS



n = 32500
Max = 20.3 ns
Min = 0.0 ns
Mean = 5.7 ns
Median = 5.5 ns

T_A = 25 °C
Sync_Interval: 2 s
Network: Switched Ethernet
Link_Status: 100 Mbit/s, FD
Oscillators: OCXOs, 5E−10
Nodes: 4, SynUTC

Measurement Results Software TS



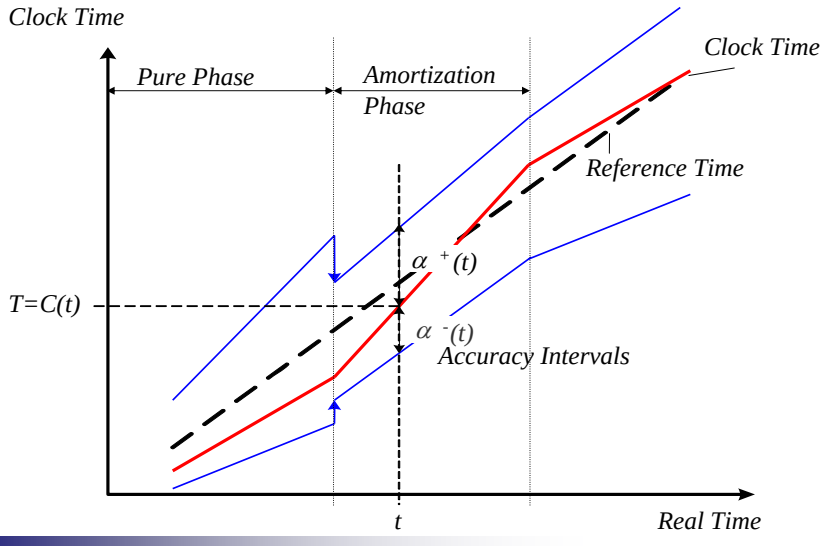
n = 60000
Max = 981.1 ns
Min = 0.3 ns
Mean = 214.3 ns
Median = 204.4 ns
Stddev = 108.2 ns
Allandev(1 s) = 0.04 ppm

T_A = 25 °C
Sync_Interval: 2 s
Network: Switched Ethernet
Link_Status: 100 Mbit/s, FD
Oscillators: OCXOs, 5E−10
Nodes: 4, IEEE1588

Conclusion

- 10 ns Accuracy requires
  - Stable local oscillators
  - High internal clock frequency
  - High speed data sampling at clock domain borders
  - High clock resolution
- This allows for low sync rates (2 sec!)
- On-Going work
  - Implement Unit into MAC-core (OPENCORE)
  - GB Ethernet design and analysis

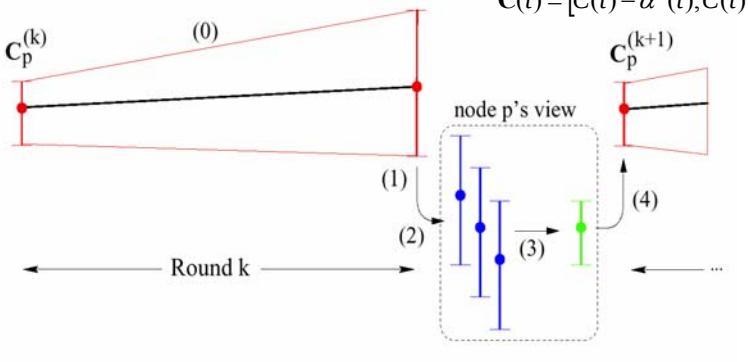
Intervals and Cont. Amortization



Re-synchronization Mechanism

- Interval clocks instead of ordinary clocks
- Local clock values are adjusted
- Accuracy intervals are adjusted

$$\mathbf{C}(t) = \left[C(t) - \alpha^{-}(t), C(t) + \alpha^{+}(t)\right]$$



HIRSCHMANN
Automation and Control

Experiences with IEEE 1588 in Higher Cascaded Ethernet Networks:

Evaluation of network elements

Hirschmann Automation and Control GmbH
Dirk Mohl

HIRSCHMANN
Automation and Control

IEEE 1588 Synchronisation precision

- Overview
- IEEE1588 over Ethernet
  - HUBs
  - Switches
  - Boundary Clocks
  - TCs
- Network reconfiguration:
  - RSTP
  - HiperRing

HIRSCHMANN
Automation and Control

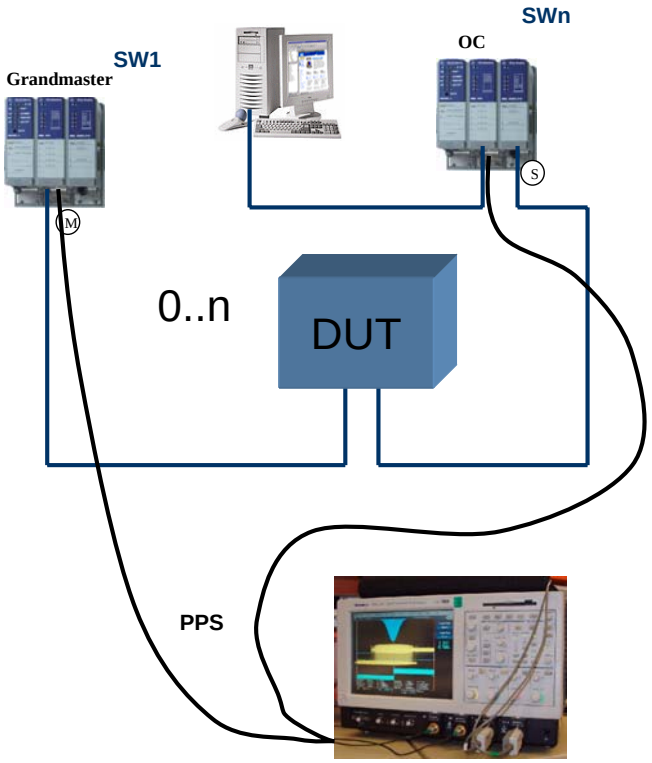
IEEE 1588 Network Design and Precision

The precision of IEEE1588 depends on the used network elements

In an Ethernet network the following connections are possible

- direct
- through a hub
- through a cots switch
- through a boundary clock
- through a transparent clock
- router / gateway ... (not covered here)

There are already many studies and measurements available here a summary



HIRSCHMANN
Automation and Control

IEEE 1588: direct connection

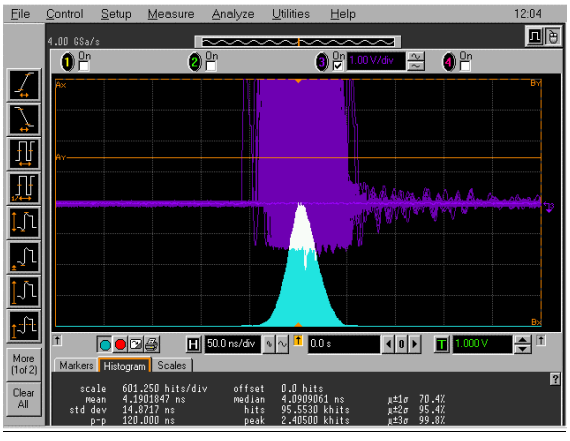
Basis for all measurements is the examination of a direct connection

Ordinary Clock connected to Grandmaster

Measurement under the following conditions

- no load
- high network load

No dependency of network load



Offset measurement

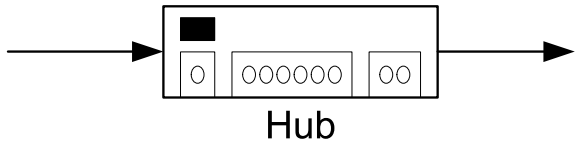
Peak to Peak: 120ns
=> Jitter: 60 ns

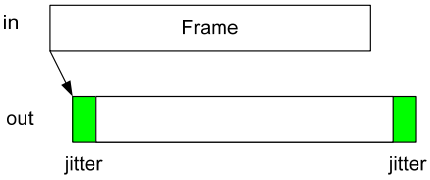
Standard deviation: 15ns

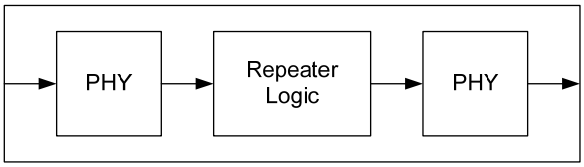
IEEE 1588 and HUBs

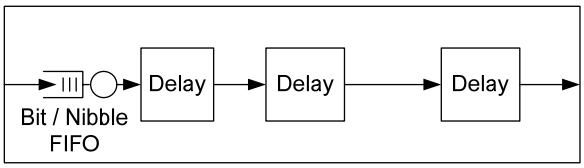
HUBs do no queuing

Bit - FIFO for incoming Data
rest is constant delay









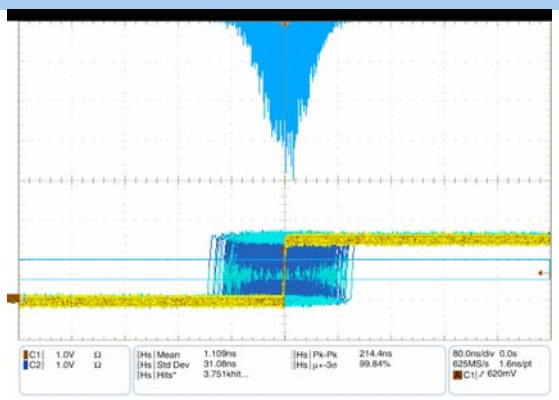
IEEE 1588 and HUBs

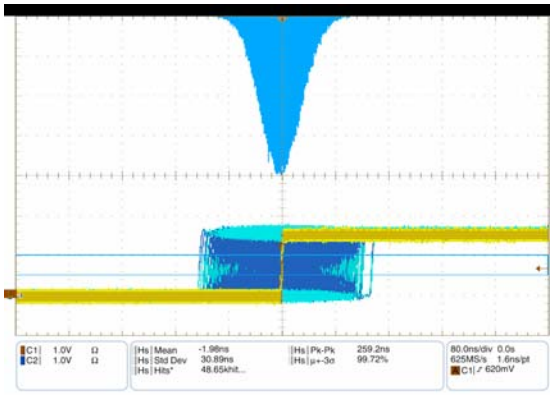
Typical HUBs

Delay: 500ns (300 ...1000ns)
Jitter: 50...100ns

Effects on IEEE1588 precision:

- no load Jitter: 110ns
- high load Jitter: 130ns

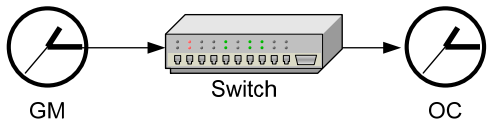


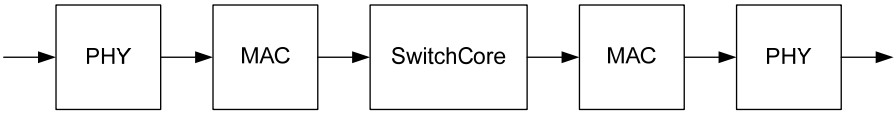


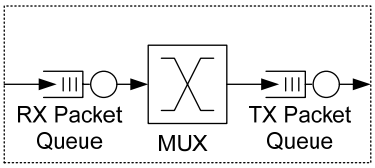
IEEE 1588 and COTS Switches

Switch

- several Bit and Byte FIFOs: delay and Jitter
- Store and Forward: packet delay
- one or more packet queues
- delay / loss of packets depending on load and queue length
- asymmetric delay



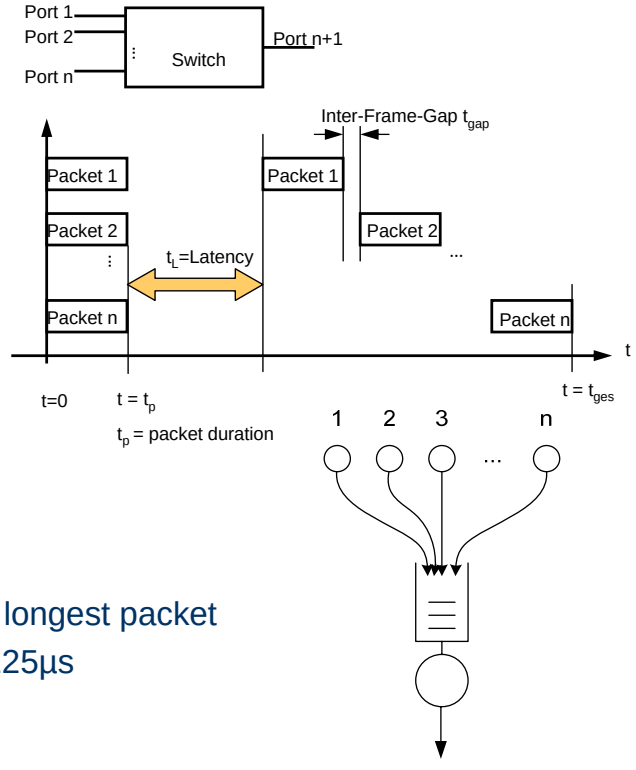




HIRSCHMANN
Automation and Control

IEEE 1588 and COTS Switches

Typical Switch



no load condition:
Store & Forward: 64 Bytes 5,6µs
Delay: 3µs (1µs ... 50µs ...)
Jitter: 500ns (200ns ... 10µs)
Asym. delay: 0... 100ns (up to ?)

with network load:
same plus queuing jitter

- worst case queue length x time for longest packet
- most cases 0 to one long packet: 125µs

HIRSCHMANN
Automation and Control

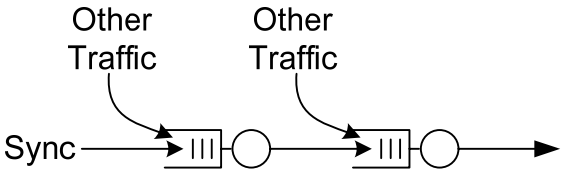
IEEE 1588 and COTS Switches

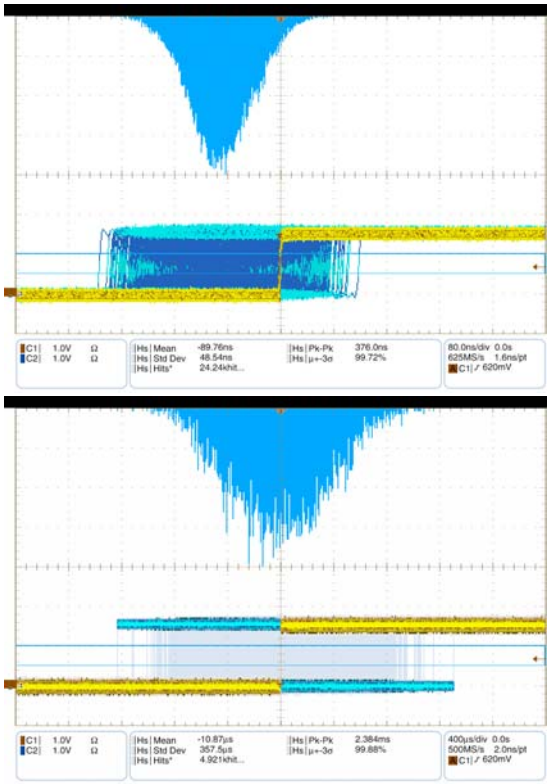
Measurement with a COTS Switch

no load
- jitter: 190ns

typical network load:
- 10% random + bursting
- jitter: 1,5ms

cascading of switches
- => cascading of jitter





HIRSCHMANN
Automation and Control

IEEE 1588: Boundary clocks

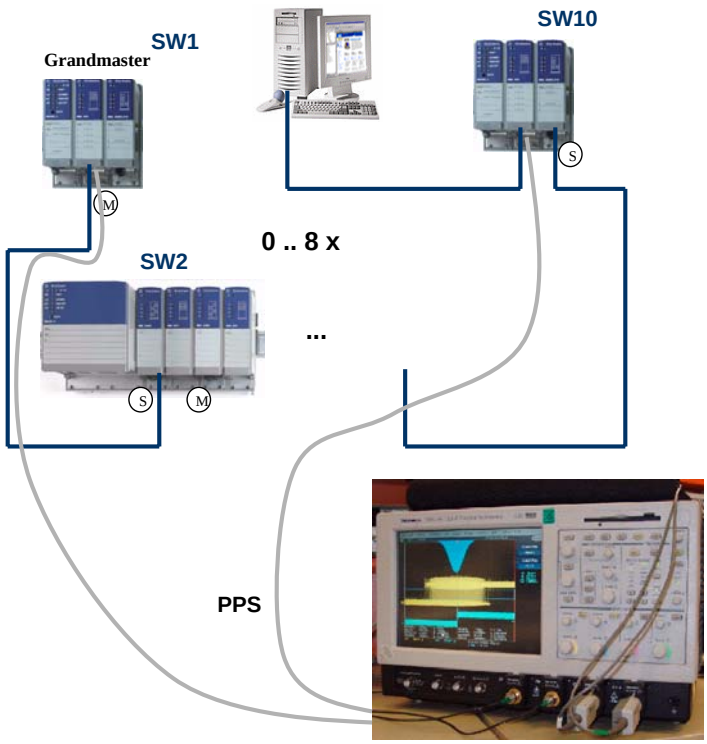
Boundary clocks

To distribute high precision in a network also the switches need IEEE1588 support

typical BC:
jitter: 50 ... 100ns

Expected behavior on cascaded BC:
addition of jitter

Nevertheless cascading of boundary clocks shows non linear effect due to control loops

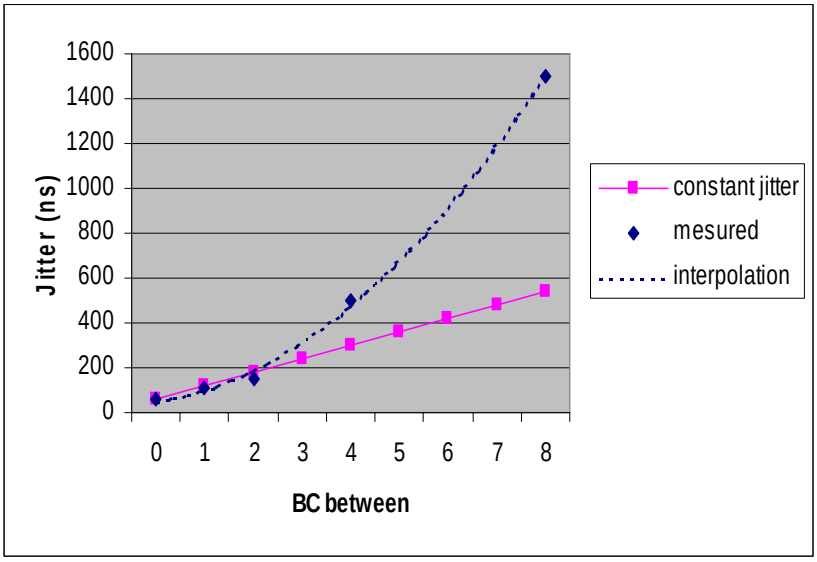


HIRSCHMANN
Automation and Control

IEEE 1588: cascaded BCs

Summary of measurements

Behavior of connecting of 2 to 10 boundary clocks



HIRSCHMANN
Automation and Control

IEEE 1588: Transparent Clocks

Transparent Clock

Constants

Sync packet time: T_{sp} = 13.1µs (164bytes at 100MBits/s)

Interframe gap: T_{ifg} = 0,96µs

Switch delay: T_{sd} = 3µs + 0,5µs Jitter

PHY Jitter + Quantisation: T_j = 30ns (10...30ns)

Free running oscillator: P_f = 100ppm
Frequency compensated clock: P_s = down to 10ns / s = 10ppb

Relevant Factor: Residence Time

HIRSCHMANN
Automation and Control

IEEE 1588: Transparent Clocks

Transparent Clock

Hardware Timestamp correction

best case: no load => no packet in queue
residence time: $T_{rmin} = T_{sp} + T_{sd} = 17\mu s$

typical worst caste max. one packet long packet in queue
residence time: $T_{rmax} = 125\mu s + T_{rmin} = 142\mu s$
=> Jitter: $T_j + (T_{rmax} - T_{rmin}) * P_f = 30ns + 12{,}5ns = 43ns$ (Free running osc.)

absolute worst case: overload queue = length x long packets in queue (ms)
=> Jitter load dependant

HIRSCHMANN
Automation and Control

IEEE 1588: Transparent Clocks

Follow UP capable Transparent Clock =>

Software Timestamp correction (but still HW- Time- stamping)

Residence time depending on SW- Stack, typical values
$T_{rmin} = 1ms$
$T_{rmax} = 20ms$

Expected Jitter:

Free running Jitter = $T_j + (T_{rmax} - T_{rmin}) * P_f = 30ns + 1{,}9\mu s$
Frequency corrected Jitter = $T_j + (T_{rmax} - T_{rmin}) * P_s = 30ns + 0{,}19ns$*
*theoretical, typical: digital adjustment +/-10ns

=> Software based TC only reasonable with additional frequency correction

HIRSCHMANN
Automation and Control

IEEE 1588 RSTP reconfiguration

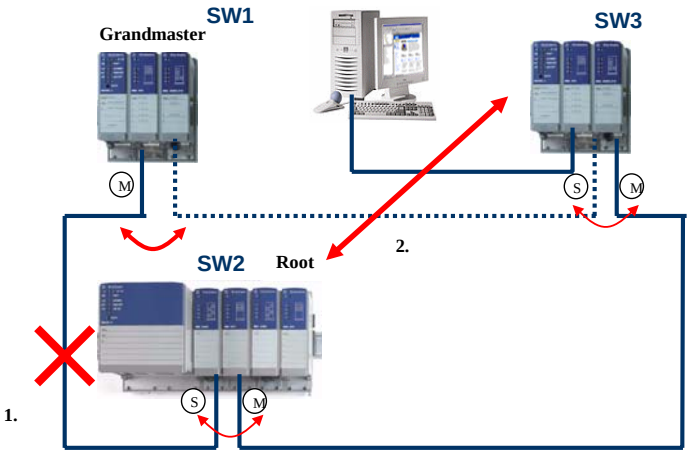
RSTP

1 Grandmaster
2 cascaded clocks

data rate: 100 MBits/s

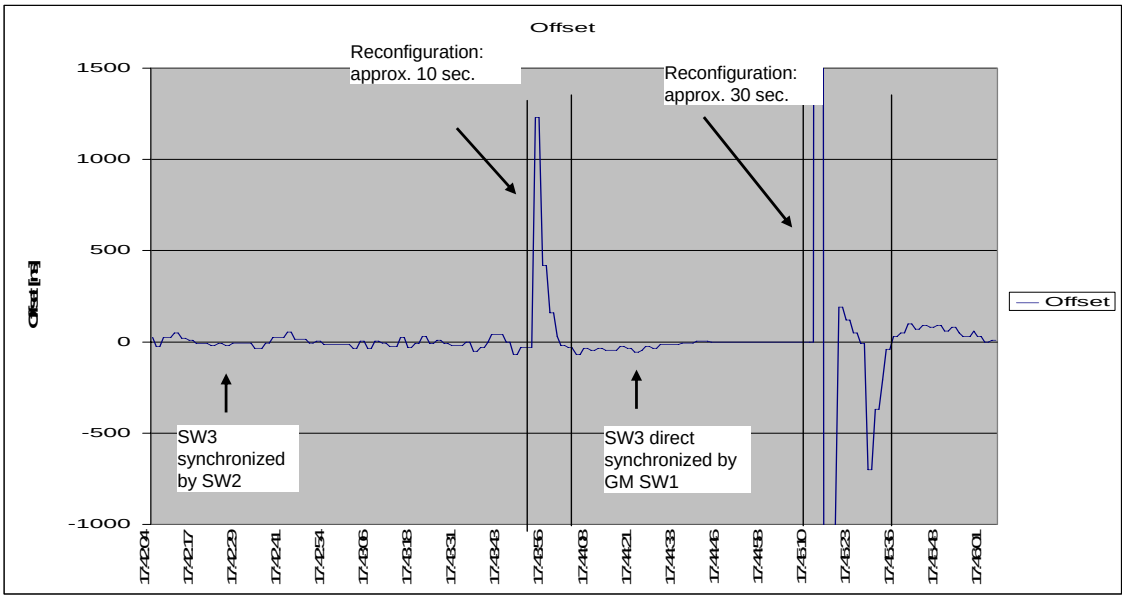
Reconfiguration time:
typ. 500ms
max. 30 s + more

Tests:
1. link break
2. root change



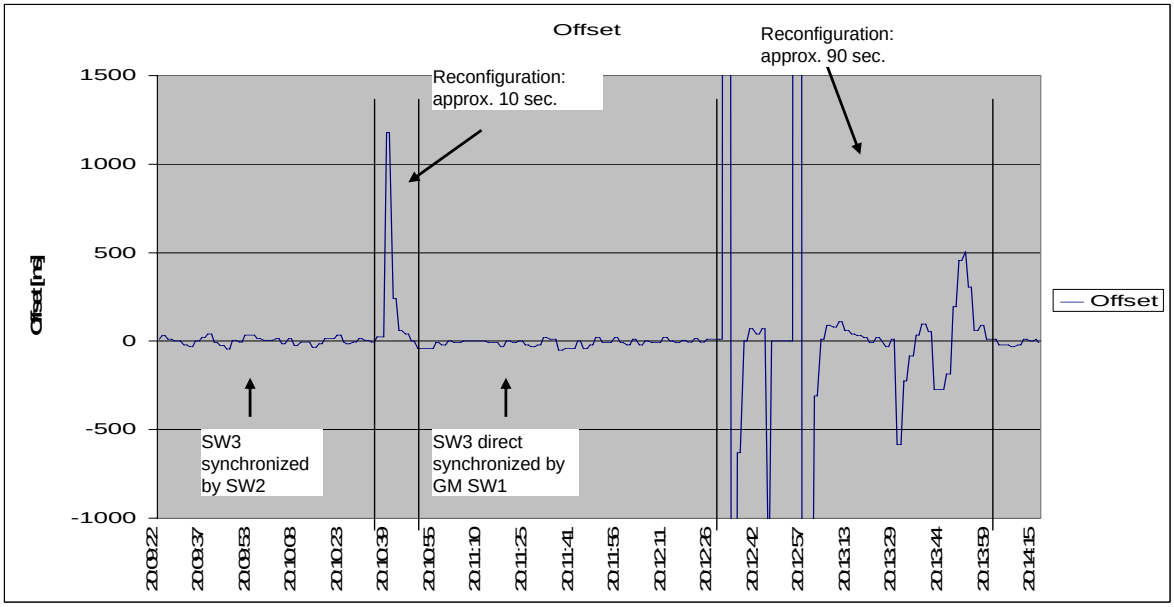
HIRSCHMANN
Automation and Control

IEEE 1588 Test results: RSTP reconfiguration



Fast RSTP Reconfiguration: Ring break

IEEE 1588 Test results: RSTP reconfiguration



Slow RSTP Reconfiguration: Root change

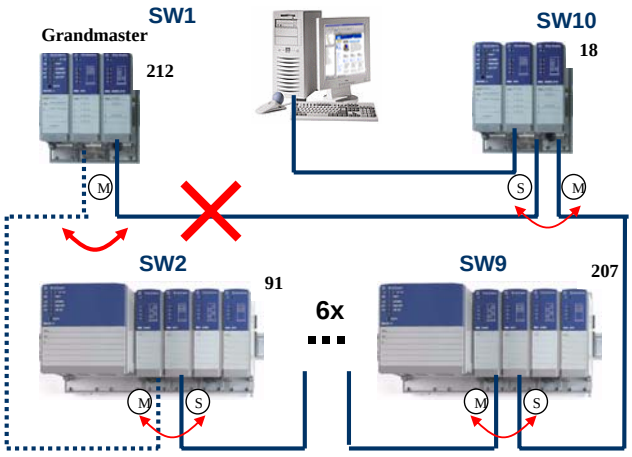
IEEE 1588 reconfiguration with HiperRing

HiperRing

1 Grandmaster
9 cascaded clocks

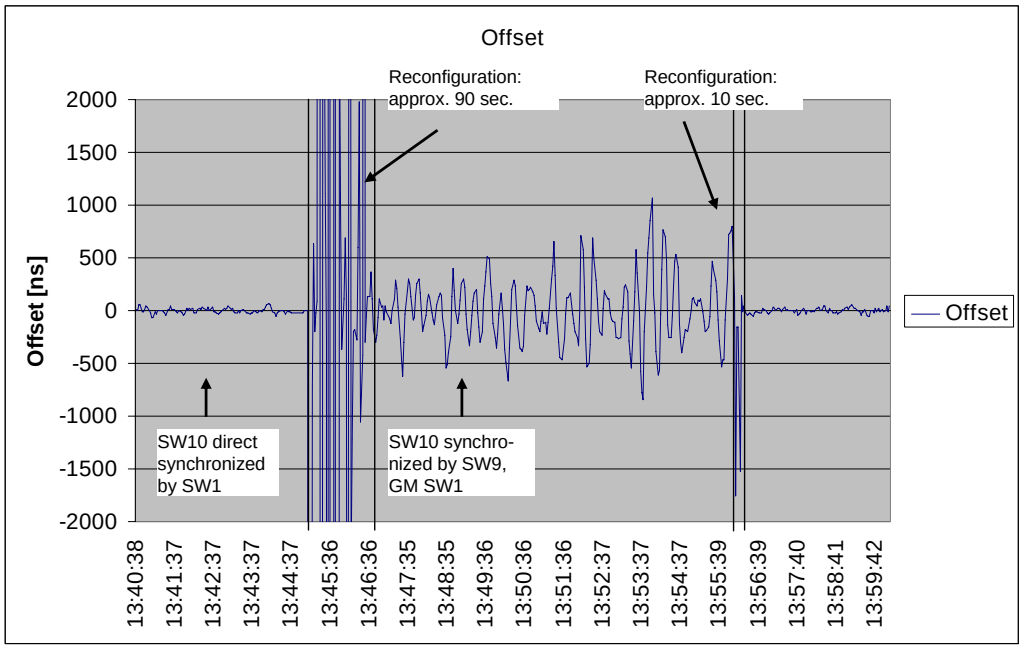
data rate: 100 MBits/s

Reconfiguration time:
typ. 200ms
max. 500 ms



HIRSCHMANN
Automation and Control

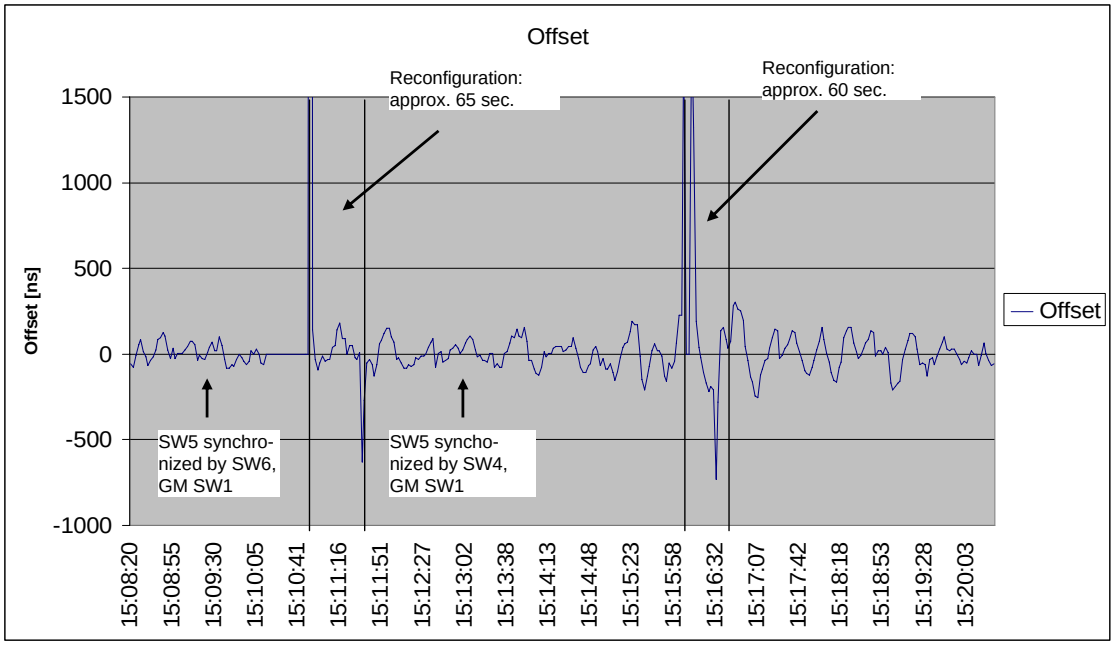
IEEE 1588 Test results: short to long path



Switch 10 and Switch 1
Fall over from direct connection to in between 8 switches and back.

HIRSCHMANN
Automation and Control

IEEE 1588 Test results: different path



Switch 5 and Switch 1
Fall over from connection over SW 10 to SW 6 to connection over SW 2 to SW 4 and back.

IEEE 1588 - Summary

Summary

- HUBs as expected nearly as good as direct connections but not suitable for high network load
- COTS switches introduce high jitter, strongly load dependant, complex statistical methods necessary do describe behavior, Precision: even in “no load condition” jitter and asymmetry has to be taken into account, Gigabit Ethernet switches will reduce effect
- Boundary clocks have problems with higher cascading depth
- TC seem to solve the problems, measurements necessary
- Reconfiguration of network currently very slow and also dependant of used redundancy protocol:
  - RSTP: depending on PTP (10s) and RSTP (90s)
  - Other: depending only on PTP

Real-Time with Ethernet

Ethernet - Just in Time

More Information:

http://ieee1588.nist.gov/
http://www.iaona-eu.com
http://www.ethernet-powerlink.org
http://www.hirschmann-ac.com

Hirschmann Automation and Control GmbH
Stuttgarter Strasse 45-51
72654 Neckartenzlingen, Germany
www.hirschmann-ac.com

Dirk S. Mohl **Dirk.Mohl@hirschmann.de**

A Distributed Test and Measurement System Application Using IEEE 1588

Alex McCarthy
Instrument Control Product Manager
National Instruments
+1 512-683-5310
alex.mccarthy@ni.com

ni.com

NATIONAL INSTRUMENTS

Distributed Shaft Twist Measuements

- Correlated measurements on large device under test (DUT) is a persistent problem
  - Big science (e.g. accelerators, field arrays)
  - Long shafts
  - Airplane wing
- Centralized measurement system
  - Limited ability to synchronize or correlate measurements
  - Signal conditioning problems
  - Long cables

ni.com

NATIONAL INSTRUMENTS

User Perspective…

F-18 Aircraft Static Fatigue Test System
Boeing

ni.com

NATIONAL INSTRUMENTS

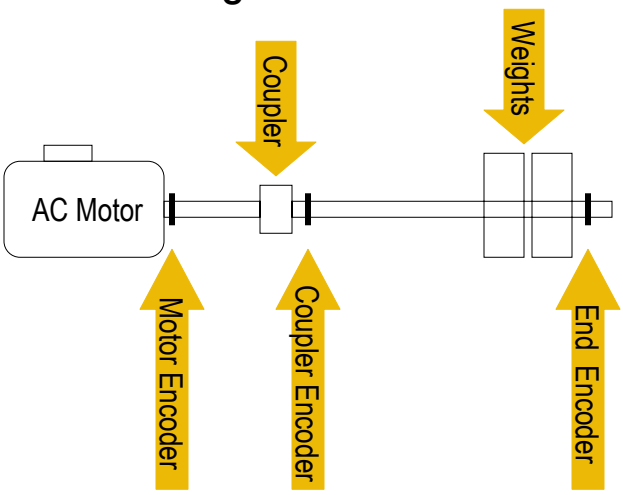
Long Rotating Shaft Model

- Develop shaft model
- Compare traditional centralized system with 1588 distributed system
- IEEE 1588 distributes synchronized clocks
- Two approaches
  - Generate local clock signal (10 KHz, 1MHz, etc.) to get counter values (time is constant)
  - Time stamp ticks (ticks or angular position is constant)

ni.com

NATIONAL INSTRUMENTS

Shaft Model Diagram

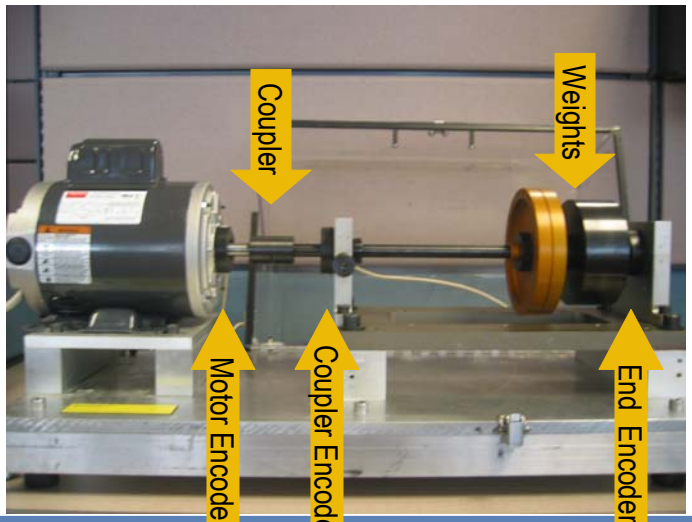


ni.com

NATIONAL INSTRUMENTS

5 2005 IEEE 1588 Conference, Zurich, Switzerland

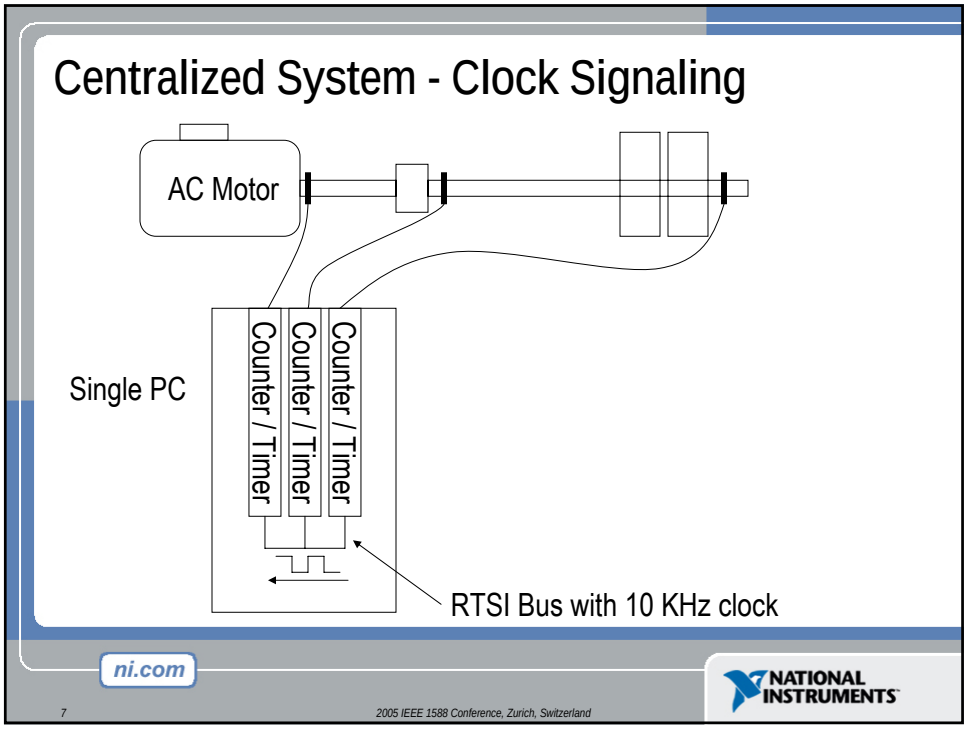
Shaft Model Picture

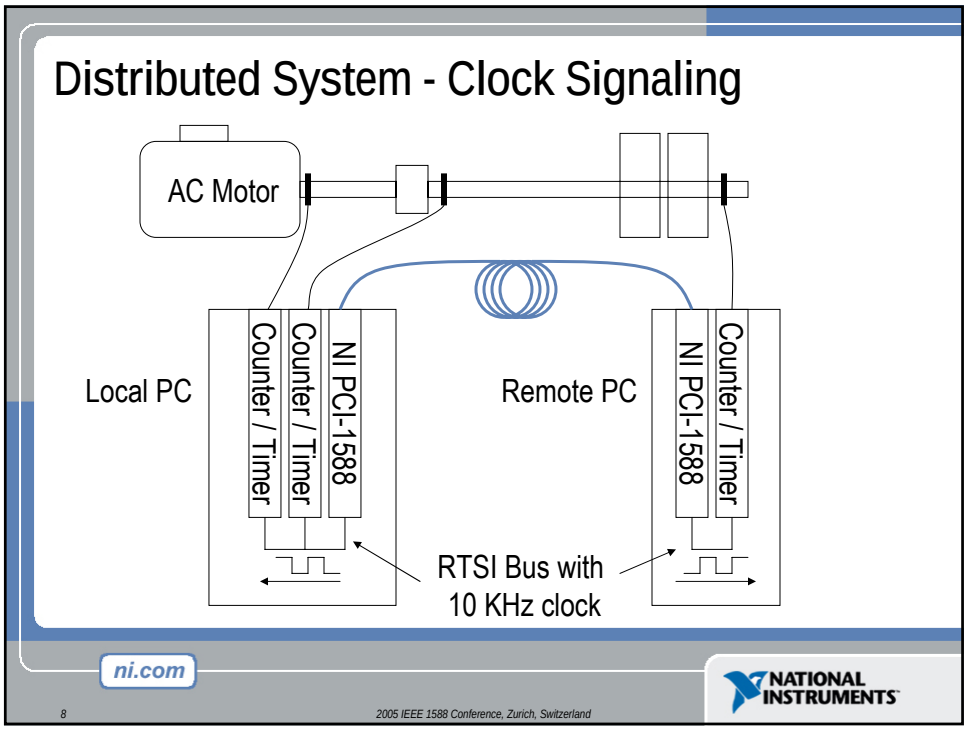


ni.com

NATIONAL INSTRUMENTS

6 2005 IEEE 1588 Conference, Zurich, Switzerland


Centralized System - Clock Signaling
AC Motor
Single PC
Counter / Timer
Counter / Timer
Counter / Timer
RTSI Bus with 10 KHz clock
ni.com
NATIONAL INSTRUMENTS
7
2005 IEEE 1588 Conference, Zurich, Switzerland


Distributed System - Clock Signaling
AC Motor
Local PC
Counter / Timer
Counter / Timer
NI PCI-1588
Remote PC
NI PCI-1588
Counter / Timer
RTSI Bus with
10 KHz clock
ni.com
NATIONAL INSTRUMENTS
8
2005 IEEE 1588 Conference, Zurich, Switzerland

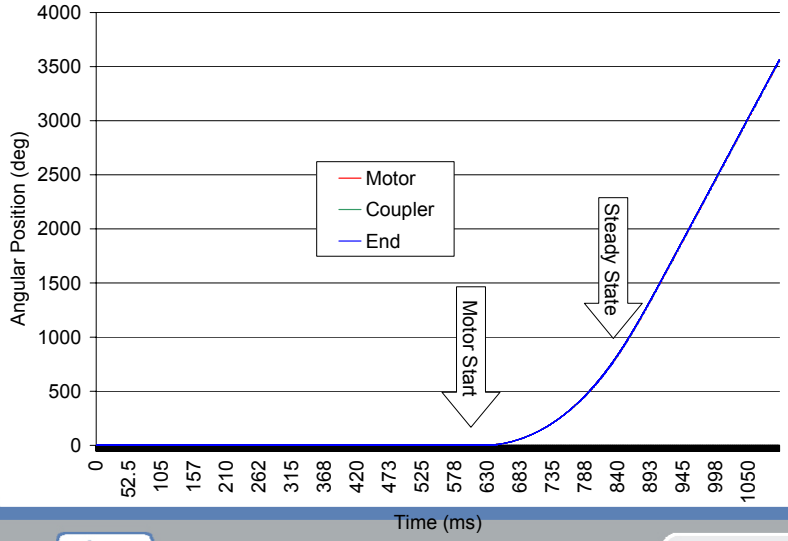
System Sizing

- Motor
  - ~2000 RPM, built-in inverter
  - Single phase, 120 V AC, 60 Hz
- Encoders: 2500 ticks per rev.
- Sizing
  - Min. detectable twist = 0.144 degrees per tick
  - Max. tick frequency = 83 KHz
  - Max. allowable clock jitter = 12 ppm (+/- 6 ppm)

ni.com

NATIONAL INSTRUMENTS

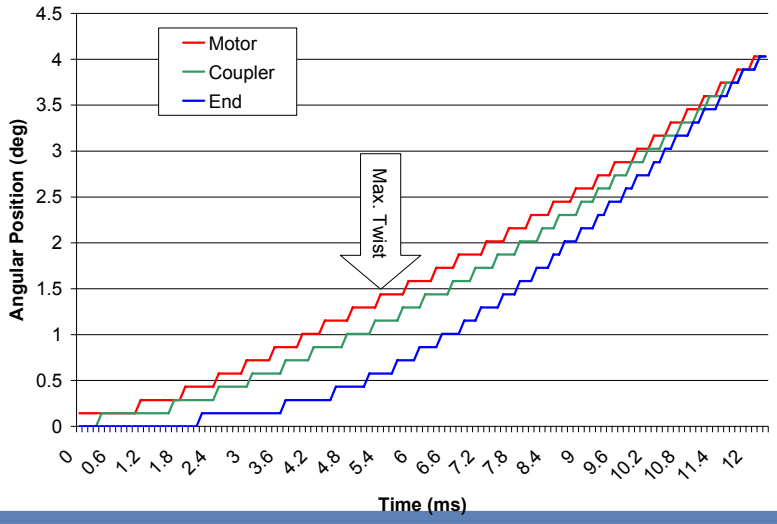
Single System – Overall Results



ni.com

NATIONAL INSTRUMENTS

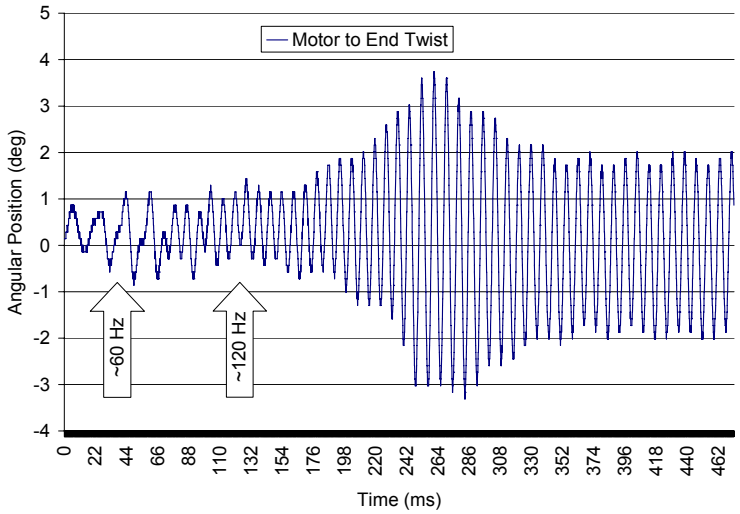
Single System – Starting Results



ni.com

NATIONAL INSTRUMENTS

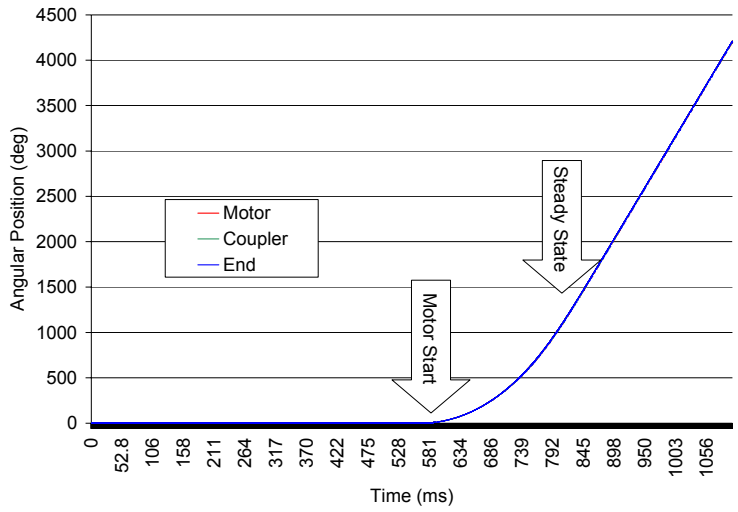
Single System – Motor to End Relative Position



ni.com

NATIONAL INSTRUMENTS

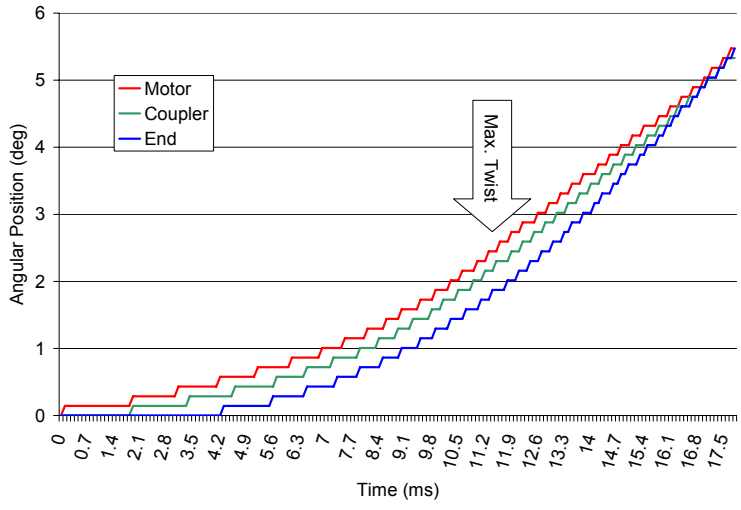
1588 System – Overall Results



ni.com

NATIONAL INSTRUMENTS

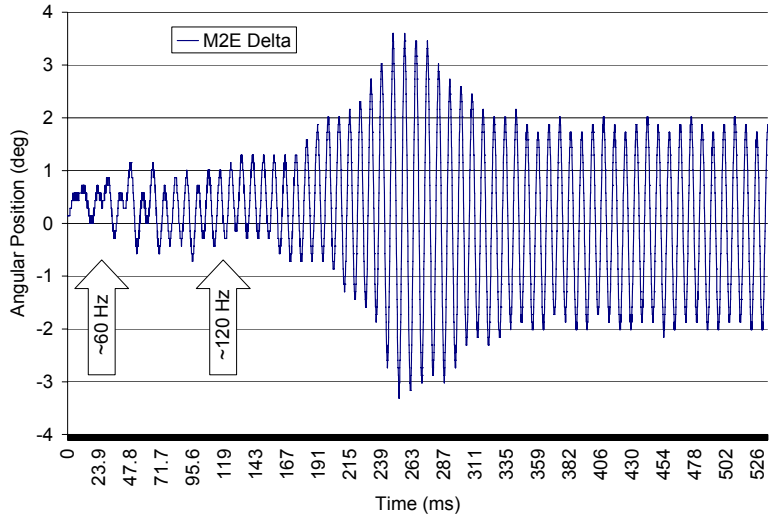
1588 System – Starting Results



ni.com

NATIONAL INSTRUMENTS

1588 System – Motor to End Relative Position

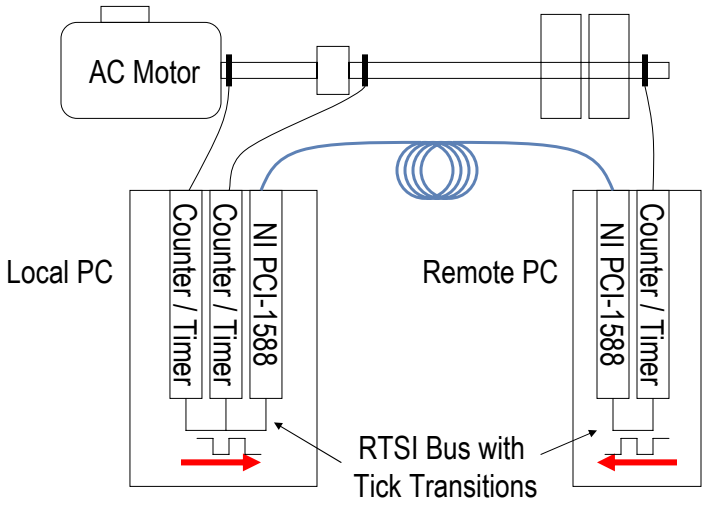


ni.com

NATIONAL INSTRUMENTS

15 2005 IEEE 1588 Conference, Zurich, Switzerland

Distributed System - Time Stamp Ticks

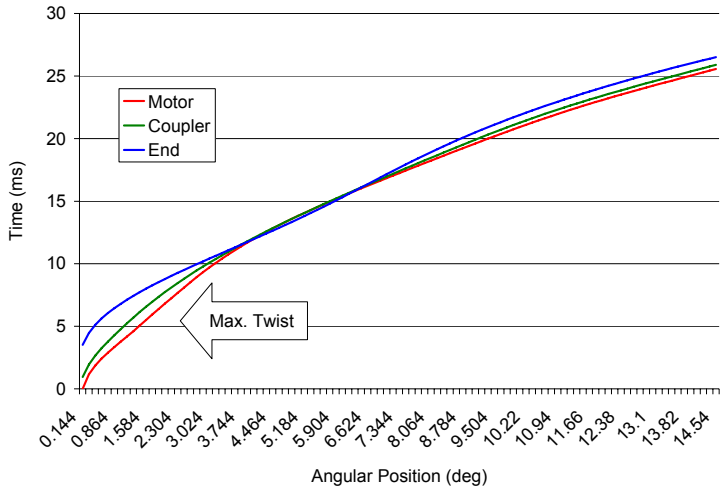


ni.com

NATIONAL INSTRUMENTS

16 2005 IEEE 1588 Conference, Zurich, Switzerland

1588 System – Time Stamp Starting Results



ni.com

NATIONAL INSTRUMENTS

Conclusions

- Both systems accurately measured twist
  - Time base jitter less than allowable clock jitter
  - NI PCI-1588 sync skew 200 ppb < 12 ppm
- IEEE 1588 provides solution to large scale measurements
  - Multiple nodes
  - Shorter signal paths
  - Longer distances between nodes

ni.com

NATIONAL INSTRUMENTS

IEEE 1588 in Test and Measurement Applications as Specified in LXI Standard v1.0

Bob Rennard
Agilent Technologies
LXI Consortium Chairman

LXI Objectives

Bring the power of LAN to the T&M industry
- Leverage the most widely accepted communications interface in the world

Simplify system integration
- Automatic discovery, addressing, network management, …
- Software environment/OS independence; identical specs and commands
- Industry-standard browsers and drivers

Lower test system costs
- Shrink overall size
- Low cost Ethernet components

Increase instrument availability
- Leverage broad industry R&D investment
- Bridge bench and systems use models
- Preserve specs and increase scale

Open new possibilities
- Peer-to-peer, remote, distributed, synthetic, simultaneous, asset management…

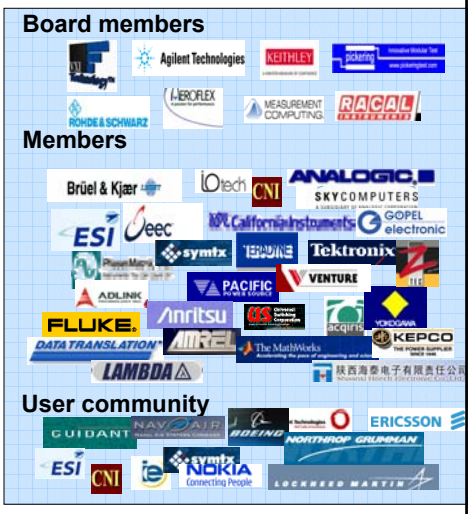
LXI Consortium

History

- Incorporated and announced Sept 2004
- First meeting Nov 17-18, 2004
- Open meetings every 8 weeks
- First plug fest May 05
- Spec released Sept 2005

Consortium Goals

- Create a standard that ensures interoperability
- Promote adoption
- Market the standard and protect the trademark

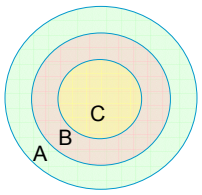


What is LXI

- LAN replaces GPIB. Common LAN implementation
- Small modules with no cardcage and traditional box instruments.
- Web browser interface for set-up, control, data access
- IVI driver API
- Hardware trigger
- Synchronization over LAN using IEEE 1588 Precision Time Protocol

What is LXI

LXI Classes

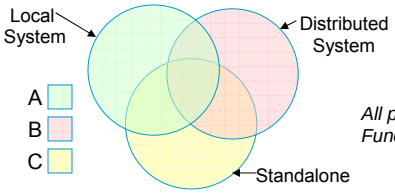


Class C: Stand alone use
- LAN replaces GPIB. Consistent LAN implementation
- Small modules with no cardcage and traditional box instruments.
- Web browser interface for set-up, control, data access
- IVI driver API

Class B: Stand alone and distributed use
- **Class C plus**
- Synchronization over LAN using IEEE 1588 Precision Time Protocol

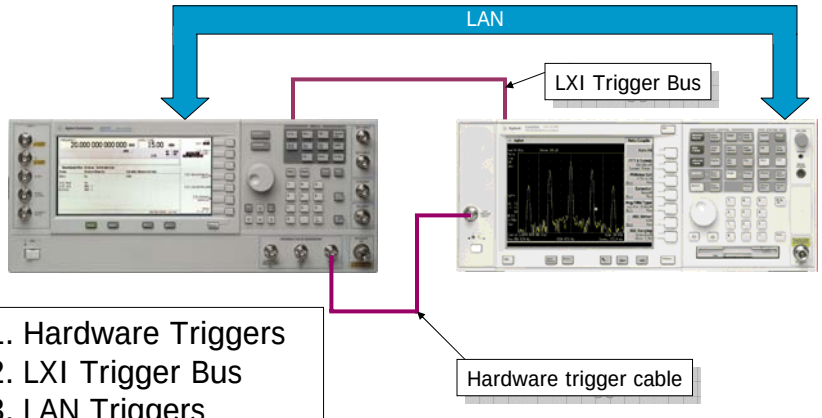
Class A: Stand-alone and any system
- **Class B plus**
- Add fast hardware trigger bus

LXI Use Cases



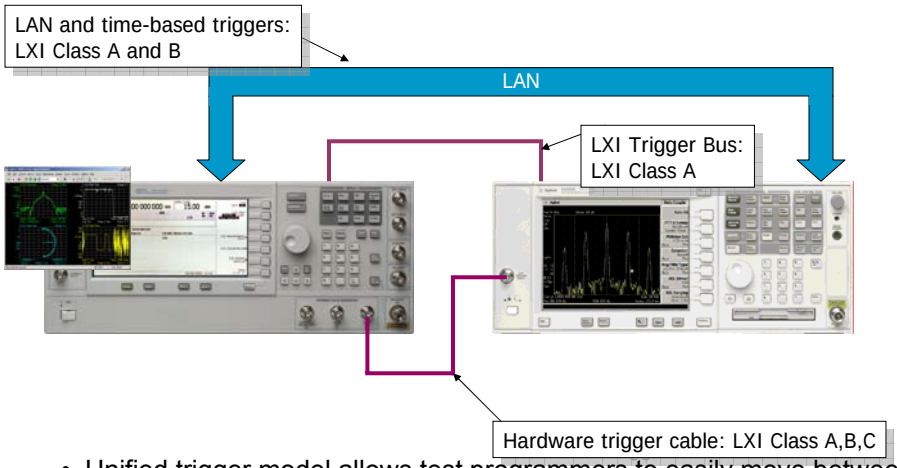
All products have consistent LAN implementation
Functional classes may be in any physical size

LXI Unified Trigger Model
Available LXI Triggering



1. Hardware Triggers
2. LXI Trigger Bus
3. LAN Triggers
4. Time-based Triggers

LXI Triggering Capabilities by Class



- Unified trigger model allows test programmers to easily move between hardware, software, and time-based triggers, simplifying software and cabling

LAN versus Hardware Tradeoffs

LXI triggering is more flexible than anything else available
Clear tradeoffs between LAN and hardware triggering

Use hardware triggers if:
- You need very repeatable timing, low jitter
- You need low latency, and the distance is short

Use Time-based Triggers if:
- You need *very* low latency (distance doesn't matter)
- You can schedule events in advance

Use LAN triggers if:
- Instruments are far apart
- You need the time stamp to look back at previously stored data
- You want to eliminate cabling from the test system

Often, it really doesn't matter
- Use the one you like better

IEEE 1588 Time Based Triggering

- IEEE 1588 tells you how to synchronize clocks over a network like Ethernet
- IEEE 1588 does not specify what to do with synchronized clocks
- The LXI specification tells you:
  1. How to use synchronized clocks in instrument systems
  2. Provides an API for IEEE 1588-based triggering

LXI Trigger/Synch applications

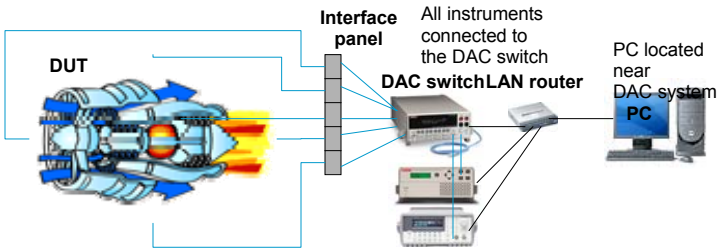
LXI Trigger/Synch enables

- Sequencing control within a module or across a system
- Control event timing
- Data and event timestamps
- Direct module-to-module communication over LAN
- Scripting

Typical applications

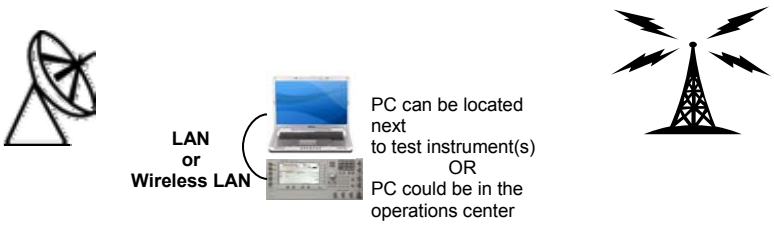
- Schedule an action in advance
- Synchronize autonomous or distributed modules
- Time stamp data from multiple channels
- Capture events backward in time

Application: Local Data Acquisition
(physical/mechanical devices)



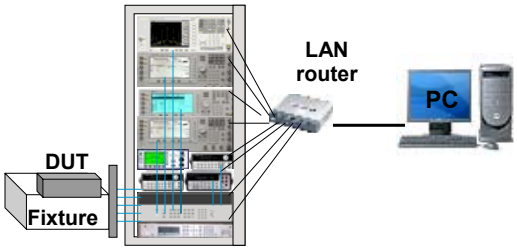
- Tens to hundreds of sensor channels to monitor or control the DUT
- Precise time synchronization between channels is often critical
- Time stamping simplifies post-acquisition analysis

Application: Electronic Signal Monitoring
(Antenna Test, Radar Test)



- Significant distances between source and receiver create latency problem with cabled systems. IEEE 1588 eliminates latency
- Location based measurements and transmit diversity measurements rely on precise time of arrival
- Time stamps simplify post-acquisition analysis

Application: Electronic Functional Test
(Manufacturing/Calibration)

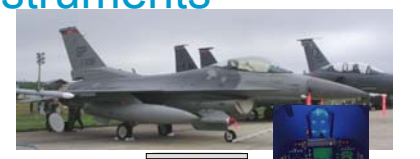


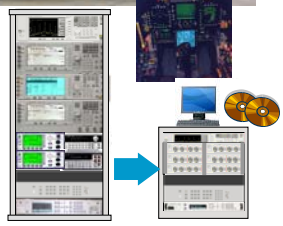
Test program controls instruments, branches and loops test, and makes pass/fail decisions. Some data is collected in archive.

- Software triggering simplifies system design, eliminates cabling
- Peer-to-peer communication between devices reduces controller traffic, speeds test throughput, and offers SW transportability
- Common sense of time simplifies instrument synchronization, eliminates hand-tuned wait statements, and simplifies support

Application: Synthetic Instruments

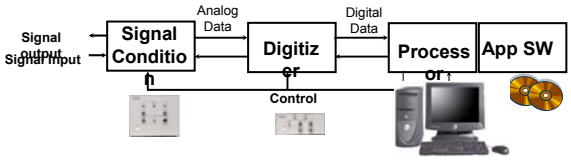
A collection of hardware and software modules combined to emulate an instrument.





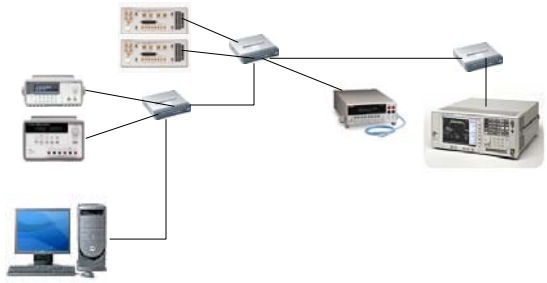
Benefits

1. Reduce size
2. Reduce Total Cost of Ownership
3. Deploy new systems faster
4. Keep systems fresh
5. Extend support life



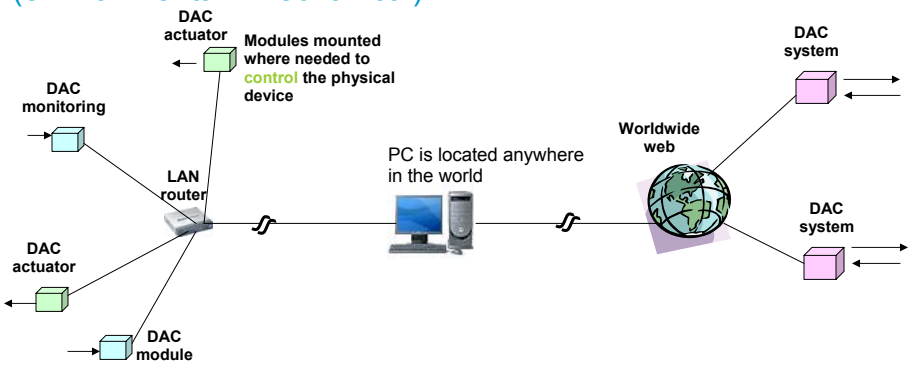
- Coordinate multiple devices
- Software triggering simplifies cabling
- Peer-to-peer communication between devices reduces controller traffic and can speed test throughput
- Common sense of time simplifies instrument synchronization

Application: Calibration / Asset Tracking



- PC searches all branches of network to discover and track instruments
- Automatically monitor assets by model, serial number, firmware revision, cal history, next cal, etc

Application: Distributed Data Acquisition
(environmental / mechanical)



- Sensors and modules are mounted where needed to monitor and/or control the physical world (temp, strain, flow, etc)
- Large number of nodes
- Distances can be significant, covering broad geographical areas

Summary: What LXI offers

LXI is based on open industry standards. No license fees
- 802.3, TCP/IP, IEEE-1588, web browsers, CAT5, IVI, standard sizes.

LXI simplifies technology migration
- Separate hardware from the application
- Unified trigger model

LXI offers new capabilities
- More engineers working on Ethernet than entire T&M industry
- Control, monitor, diagnose from across the lab or overseas
- Peer-to-peer, remote, distributed, synthetic, simultaneous, …

IEEE 1588 is a key enabler

LAN Triggering

How is LAN triggering different?

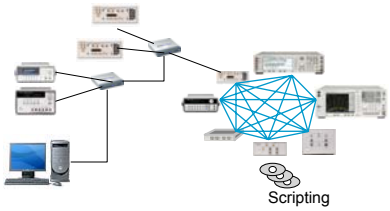
- **LAN triggers carry a time stamp**
  - Tells when the original event really happened
  - Can be used to capture events backwards in time
- **LAN triggers can be point-to-point or multicast**
  - If broadcast, instruments are programmed in advance
  - If point-to-point, system configuration is different
  - UDP latency (multicast) much less than TCP (point-to-point)
- **LAN triggers have a name**
- **LAN triggers subject to LAN traffic patterns**
  - Latency, timing jitter, possibility of loss

How is LAN triggering not different?

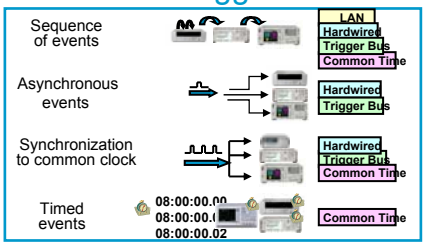
- **Unified trigger API treats all triggering methods equivalently**
  - Publish/subscribe design pattern
- **Trigger sources identified by name**
  - Spec defines “LXI0” thru “LXI7” for trigger bus
  - Spec reserves “LAN0” thru “LAN7” for LAN
  - Other LAN names user-defined
- **Example:**
  - Trigger an arb on LXI Trigger Bus line #2: Arb.Trigger.Source = “LXI2”
  - Trigger an arb via LAN trigger with name “LAN2”: Arb.Trigger.Source = “LAN2”

The future of test: LXI benefits

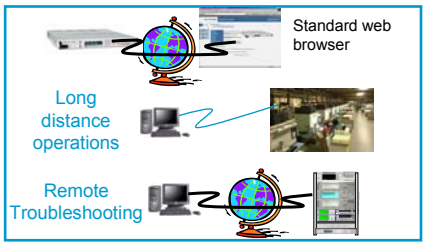
Scripting and peer-to-peer comm



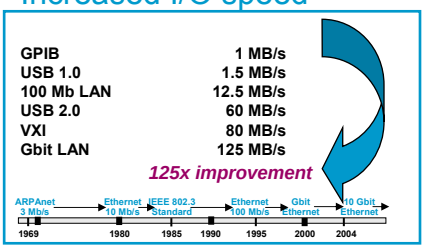
Unified Trigger Model



Remote access via Web



Increased I/O speed

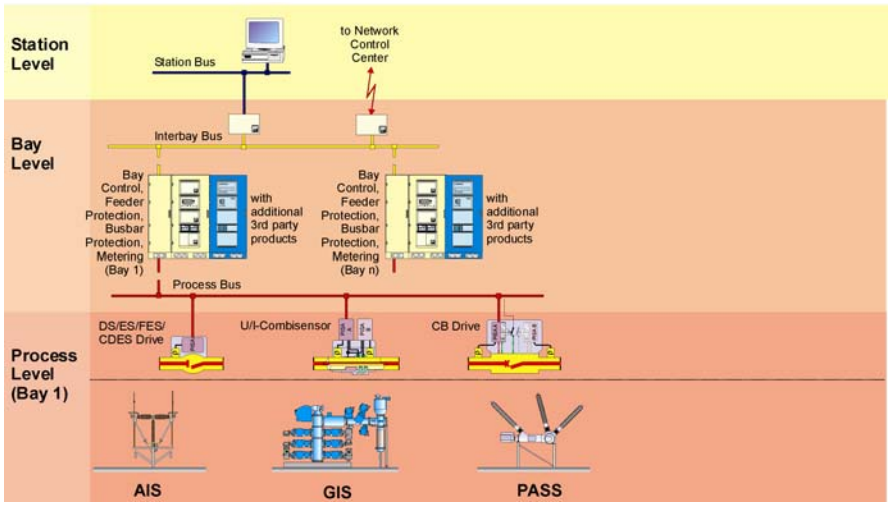


Lars Andersson

Application of IEEE1588 in Substation Automation

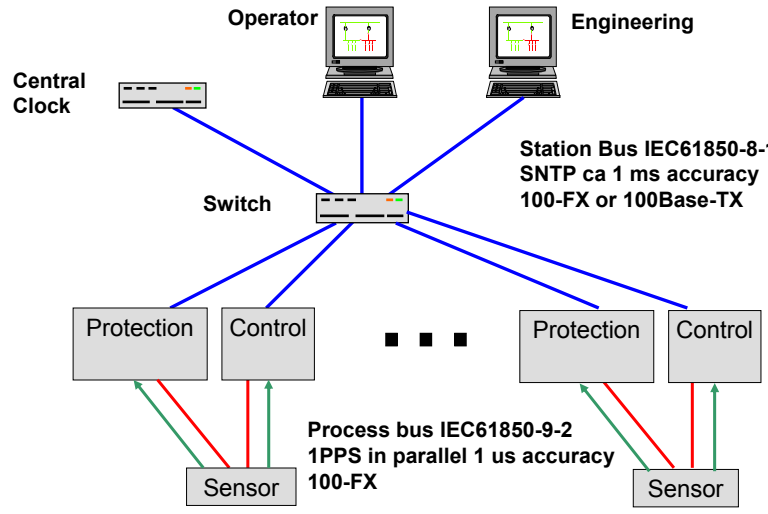
© Copyright 2004 ABB.
All rights reserved. - 1 -
07/12/2005

Substation Overview



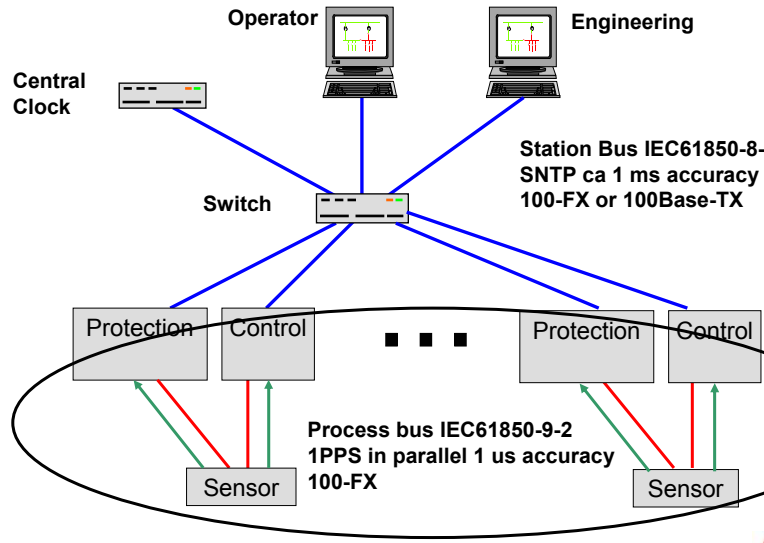
© Copyright 2004 ABB
- 2 -

System overview



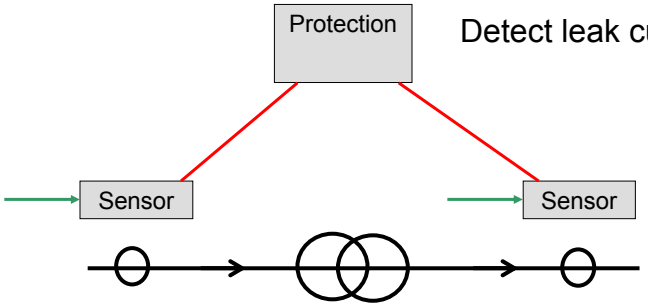
© Copyright 2004 ABB

Process level with sampled analogue values



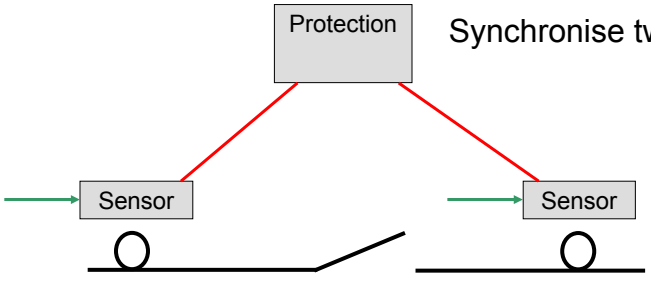
© Copyright 2004 ABB

Application: Differential currents



© Copyright 2004 ABB

Application: Synchronisation



© Copyright 2004 ABB

Allowed Maximum Phase Inaccuracies (Jitter)

- Station wide functions
  - Between any two u- or i-signals within the substation
  - Synchrocheck: ~ 1400 µsec (~ 30 degrees)
  - Differential protection: 25 µsec (Busbar and Trafo)
- Feeder oriented functions
  - Feeder oriented u-/i-vector calculation
  - Line protection: ~ 90 µsec (~ 2 degrees)
  - Metering class 0.2: < 7.7 µsec (acc. IEC60044-8)

© Copyright 2004 ABB

Summary for sensors on process level

- One microsecond accuracy between sensors as well as sensor and control & protection
- No reference to absolute time is required
- Not all devices need to be synchronised
- No central clock is wanted due to cost and availability
- Synchronisation on process level described here has nothing to do with the synchronisation required on station level
- HV Protection is always redundant
- 1PPS always require separate fibre

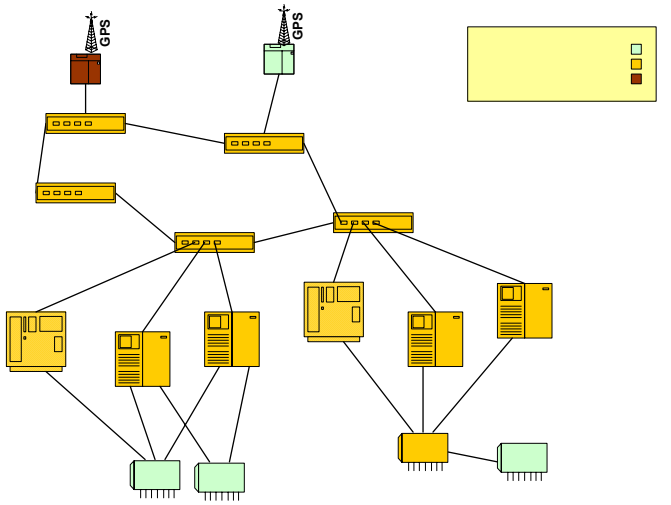
© Copyright 2004 ABB

How to use IEEE1588 with IEC61850

- Project together with Institute for Embedded Systems at Zürcher Hochschule Winterthur
  - Verify the accuracy requirements for the Ethernet architectures used in Sub Station Automation
  - Define scenarios for high availability (redundant) and graceful degradation
  - Suggest the IEEE1588 as synchronisation method for higher accuracies in connection with IEC61850
  - Prototype clock on a HW used in substation automation
- Some different scenarios:

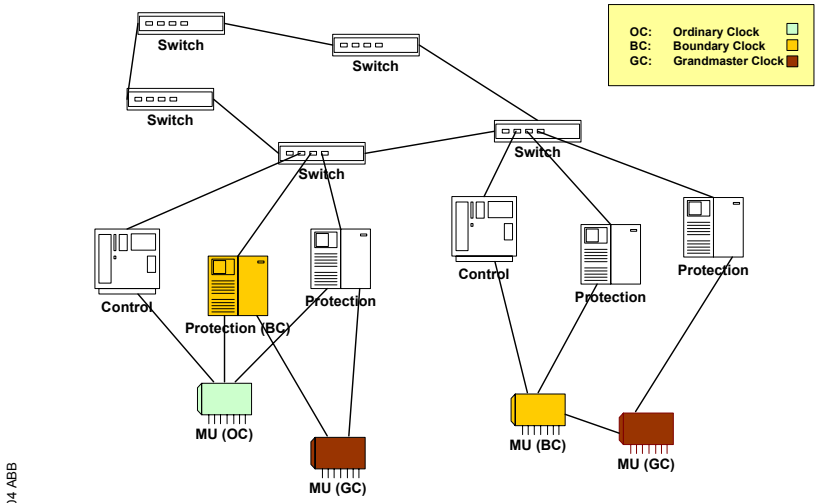
© Copyright 2004 ABB

Central dedicated GC



© Copyright 2004 ABB

GC in the sensor



© Copyright 2004 ABB

In Search of the Key to the Lock:
Clock Synchronization Issues and Requirements in Semiconductor Manufacturing

Authors: Ya-Shian Li (NIST), John Messina (NIST)

October 17, 2005

Abstract:
Precision time synchronization requirements lie on the horizon for the semiconductor manufacturing industry. Future advanced process control applications will require a common time base including the capability to rapidly merge semiconductor factory and process data from heterogeneous sources for providing intelligent data analysis in process control and equipment diagnostics. This paper will discuss the potential semiconductor applications for time synchronization and specific industry issues and requirements of semiconductor manufacturing. How IEEE 1588 potentially applies in the industry's distributed clock synchronization solution scheme and some possible hindrances to its adoption are also investigated in this paper.

Introduction

As manufacturing complexity rises with the advancement of integrated circuit technology, the semiconductor industry is striving for greater precision and automation. In implementing manufacturing automation mechanisms, the industry has discovered short comings in one of the most fundamental measurement capabilities – time. Historically, time has been a dispensable component in semiconductor manufacturing environment. Moreover, most applications did not require time to be synchronized. With the increased need in advanced automation, time has become an advantageous and even necessary component of process data for fully exploiting valuable information taken from various points within the manufacturing environment. Providing an infrastructure of accurate time throughout the factory, equipment, and each device producing data, would enable greater precision in manufacturing automation and ensure the efficiency necessary for manufacturing semiconductors at improved quality, yield and decreased cost.

To continue the advancement of Moore's Law in a commercially viable manner, the semiconductor industry must continuously push the limits of manufacturing. The current Moore's Law states that the number of transistors on a chip should double every two years. Additionally, Moore's Law coupled with periodic node shrinkages entails lower tolerance windows for each chip manufacturing process. Steadily increasing manufacturing complexity is driving the need for finer manufacturing controls in order to maintain the same yield and quality necessary to ensure the chips are commercially viable.

One example of lower tolerance windows is gate leakage [1]. As devices shrink and transistor density increases, the proximity of the transistors can cause side effects to occur, result in unpredictable chip behavior. One of the factors contributing to gate leakage is the thickness of the oxide; when the thickness is outside the tolerance range, the circuit will experience a current when it is not intended. Such effect can be disastrous to the function of the chip. Manufacturing must be carefully monitored through advanced measurement capabilities and controlled accordingly to ensure each chip manufactured is as close as possible to the intent of the design.

In order to face the challenges of increasing processing complexity, the industry has been experiencing a new wave of automation requirements. The industry has been discussing and implementing e-Manufacturing capabilities in order to automate manufacturing for consistently higher yields and quality, while ensuring the factory is running at maximum efficiency. E-Manufacturing encompasses both Advanced Process Control (APC) and e-Diagnostics. In meeting industry demands for greater data analysis capabilities, APC suppliers are finding their analysis tools are being limited by the inaccurate time stamps resulting from unsynchronized clocks.

Automated scheduling and dispatching is another area where time synchronization is essential by allowing wafers to smoothly run from tool to tool, while minimizing wait time. Because of the increasing reliance on automation, accurately clock synchronization will be required at varying accuracy requirements to ensure each part of the processing is executed with precise control.

Integrating distributed clock synchronization protocols will not be a trivial task in the chip manufacturing environment. Some factories run the fabrication equipment with little regard to the synchronization of their clocks. Therefore equipment suppliers and equipment module providers often select the least expensive timing components available without guaranteeing an interface to synchronize it. By compromising on clock quality, the vendors are able to conserve resources for building competitive advantage in direct processing capabilities. The historical negligence towards tool timing functions has led to an equipment architecture and factory environment not amenable to distributed time synchronization.

The industry is beginning to see a need for change. Precise timing will soon be an advantageous factor in advancing processing automation. However with the limited resources available the solution will must be cost-effective yet sufficient to meet the needs in order to gain industry acceptance and adoption. With its focus on industrial automation environments, the IEEE 1588

standard provides a potential distributed clock synchronization solution that meets current requirements and provides the flexibility for more stringent needs as they arise in the future.

Semiconductor Applications

Increasing reliance on automation and manufacturing precision are the key motivations driving the effort for accurately synchronized time. The three main applications driving increased timing synchronization are advanced process control, advanced data measurement and collection, and next-generation automation capabilities. Currently, the yield and quality of the final product requires a period of ramp-up time before the product becomes commercially viable. In order to meet time to market demands, the industry needs to improve their understanding of processes and the impact of different processing and tool parameters on the final chip product.

Advanced Process Control

APC allows factory engineers to monitor and control wafer or lot processing in real-time to enhance the final product yield and quality, while reducing the number of faulty wafers and dies. Such understanding of the process will require the ability to collect and merge data from heterogeneous data sources in order to perform multivariate analysis to determine cause and effect relationships.

Within the realm of advanced process control, fault detection classification and process optimization applications would benefit from time synchronization improvements. Currently most APC developers and factory engineers would like to see the distributed time synchronization accuracy between 10 ms and 1 ms.

Fault detection classification (FDC) provides the capability to detect a process excursion and be able to diagnose the fault and severity in real-time based on simple and advanced analysis on data from relevant sources.

Process optimization is the capability to refine a new process ramp-up by improving the yield and quality of the chips. Process optimization ensures a process is able to produce product wafers at peak performance. Process optimization will also require a precise understanding of all the parameters involved during chip fabrication and their complex inter-relationships.

APC will require precision time stamping in order to merge all of the potentially relevant sources involved in the wafer production such as sensor data, process parameters, tool data and factory environment. Through the collective use of available data, factories can perform more complex multivariate or other advanced correlation and analysis in order to expose new cause-effect relationships for improved control as they strive to optimize the process. The time stamps will allow events occurring before a process anomaly to be properly ordered for rapidly tracing to the root cause.

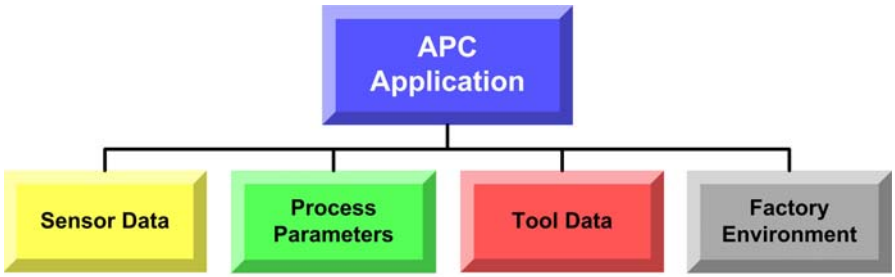


Figure 1 APC applications require the merging of data from heterogeneous sources to precisely determine new cause-effect relationships.

An example is Chemical Vapor Deposition (CVD) where gas is released to dope the surface of the wafer. The CVD process requires a wafer to be placed inside a chamber of a CVD tool and exposed to a precise combination of gases to create an ionized wafer surface with semiconductor properties. The ability to measure data and control the deposition process as the wafer is undergoing CVD allows the industry to produce ultra large scale integrated circuits while utilizing materials which can be difficult to work with due to their increased chemical complexities.

As engineers are adding more transistors into decreasing device sizes, the deposition process requires more precise knowledge of the gas chemistry inside of the chamber as well as the evolving surface chemistry of the wafer over time. The CVD process step requires knowledge of chemical nature, concentration, and physical distribution for process modeling which can be obtained from various metrology tools. Control of temperature, gas flow, and pressure can all affect the final product [2]. Having accurate time stamps on each piece of data arriving from multiple tool sources would facilitate the merging of the various pieces of data. The ability to analyze data from heterogeneous sources enables the learning of the exact process parameters and environmental conditions necessary to improve semiconductor manufacturing processing.

Advanced Measurement

Measurement capabilities are imperative in APC for effectively and precisely controlling processes. Metrology tools using sensors to deliver data at consistent rates while a wafer is being processed provides especially insightful information on process excursions, which can adversely impact a wafer. Conversely, when a wafer produced in a specific flow has a high quality and yield, it behooves the factory to be able to duplicate the processing environment as precisely as possible. In-situ measurement ensures the deviations can be detected quickly and process can be adjusted immediately to rectify the situation. Advanced measurement capabilities include the ability to perform measurements while a wafer is being processed. The measurements should yield quality data with accurate time stamps in order to ensure the data can be merged with other data sources in the tool or factory environment. Measurement is becoming critical to the processing, and in turn the industry has developed the standard equipment interface that supports data collection rates of up to 10 kHz. Disciplined, synchronized clocks can be used to ensure the data collection rates are consistent within the tool and sampled as often as requested.

Such metrology tools and the sensors within the tool should be synchronized to the equipment and other equipment modules in order to allow metrology data to be accurately time stamped such that APC tools can exploit full analysis capabilities. Time stamps provide a rapid method for data to be analyzed in real-time or after a wafer run. Accurate time stamps can pinpoint when a process failure occurs and immediately take actions to trace the cause of the excursion in order to rectify the faults during wafer processing before the wafer or lot becomes unsalvageable.

Other Advanced Automation Capabilities

Other next-generation factory automation applications can also benefit from synchronized time, but the requirements are currently less stringent. For e-Diagnostics and scheduling applications, the current demand for timing accuracy is about 1 s in order to sequentially order events occurring before a fault or to schedule a lot to arrive at a specific tool. E-Diagnostics requires accurate timing for determining the root cause of a tool or processing error. Reducing tool wait time is a factor in maintaining factory scheduling efficiency. To properly ensure the factory maximizes its productivity, proper scheduling and dispatching of tools and wafers will also demand improved timing synchronization.

E-Diagnostics involves the ability to remotely monitor, maintain, diagnose and repair equipment on the factory floor in real-time. Industry e-diagnostics guidelines require time stamps originating from the equipment to be closely synchronized with the host to enable rapid correlation with other equipment events occurring in various equipment hosts, subsystems, and modules [3]. Repairing a piece of equipment may require a global network of suppliers and the ability to monitor the situation remotely conserves both time and financial resources. All modules should have

synchronized clocks in order to monitor the equipment system more effectively and to enable rapid tracing of events to for diagnosing the initial event triggering a tool excursion.

Scheduling and dispatching factory equipment can also be optimized with accurate factory clock synchronization for increased efficiency. A fabrication facility typically has dozens of process flows occurring simultaneously and with each lot at various stages of production. Each flow (lot) can have anywhere from 200 to 500 processing steps. The scheduling tool must orchestrate information from the enterprise level, manufacturing equipment systems (MES), equipment, and lot level in order to determine the optimal factory efficiency. Wafers must be transported efficiently between pieces of equipment; synchronization of the equipment would ensure a smooth transition and reduce waiting due to late arrivals if the previous tool processing the wafer were not sufficiently synchronized with the next tool. When clocks between the automated material handling system and the tool are off by minutes, a tool would be idle while waiting for the lot to arrive. Aggregated tool idle time can incur significant financial losses for a chip manufacturer [4].

Needs

Factory process engineers and APC software suppliers are discovering the limitations of APC capabilities due to the lack of synchronized time stamps attached to the collected data, the ability to consistently obtain needed data, and the need to synchronize data from multiple sources. In order to improve the data analysis, the industry will need better time stamping capabilities and to obtain data at consistent sampling rates. For application requiring the merging of data from various sources, the disparate clocks will need to be synchronized to ensure it shares a common factory time. Sampling data at high, consistent rates requires the clock frequency to be disciplined on the order of the sampling rate or better.

Poor clock synchronization in the distributed factory system makes it unsuitable to perform advanced analyses where multiple variables from different sources must be examined in order to determine the optimal combination for a particular process. Current FDC requirements are on the order of 10 ms to 1 ms time stamp synchronization, but with Interface A supporting capabilities of up to 10 kHz data sampling frequency, the requirements would fall below 1 ms to fully support the industry standard.

The ability to verify the clock synchronization accuracy is also beneficial for determining the time stamp quality associated with a piece of data. Because modules and equipment are supported by multiple vendors, the ability to verify clock stability and synchronization quality provides the factory a mechanism to ensure the data received has a time stamp uncertainty within its required tolerance. Factory systems should be able to obtain clock synchronization statistics, such as estimated uncertainty, from the synchronization protocol or the clock itself.

Another need for distributed clock synchronization in the factory is security. The clock synchronization protocol should not introduce a point of vulnerability. For instance, equipment clocks should ensure the time coming from the server is indeed a trusted master clock. In addition, both the chip maker and equipment vendors heavily prize their intellectual property and would prefer to have all the security provisions necessary to ensure their proprietary information will not be jeopardized.

Integration Challenges

Existing factory environments are typically comprised of heterogeneous equipment from multiple vendors. Devices such as sensors, actuators, metrology tools and controllers generally communicate via standard industrial protocols such as DeviceNet, Ethernet/IP, and Profinet. Equipment, equipment host, and factory manufacturing systems typically communicate via Ethernet in modern systems. The industry also has legacy systems relying on RS232, Modbus, Token Bus (IEEE 802.4) and Token Ring (IEEE 802.5). Within the equipment, the architecture can vary greatly depending on the support of the equipment supplier for industry standards to connect devices using a common device network versus a proprietary network. Currently the ability for devices to readily interoperate on a common network does not always exist.

As depicted in Figure 2, current factory time is typically obtained from a Network Time Protocol (NTP) time server synchronized to a factory time server. Routers synchronized to NTP time are also used to propagate time to factory computing systems including the equipment host. Equipment currently does not directly synchronize with NTP, it obtains its time through the equipment host using an industry specific messaging protocol, Semiconductor Equipment Communications Standard II (SECS II), sitting above TCP/IP on the OSI network stack. Advanced factory networks have very low latencies, which allow the ability to use NTP capabilities in a manner sufficient to factory needs.

The quality of the synchronization starts to degrade when the equipment host uses SECS/GEM's get and set time functions to synchronize the equipment. SECS/GEM does not have the features of a time synchronization protocol to accurately estimate the actual time when the time message is received by the equipment. The equipment directly sets the time dictated by the host, and neglects the aggregated latencies of the processing and transport time of the message.

Synchronization within the equipment poses the greatest challenge. Semiconductor manufacturing equipment is based on a modular architecture comprised of subsystems and cells provided by the equipment supplier or third party vendors. The modules provided by third party vendors are not necessarily transparent to the equipment suppliers, much less the factory. Some factory clocks within the modules are opaque, with no method to access the clock directly for synchronization. Off-the-shelf modules also do not adhere to any uniformity in time stamping format or clock interface for allowing external synchronization controls.

Lower level devices, such as sensors, typically do not have clocks and currently must time stamp the data until they reach the point where sensor data is converted from analog to digital format or when it arrives at the data collection client. Even if the clocks are synchronized the time stamps do not reflect the actual time when the data was generated. These issues all contribute to poor time stamping accuracy.

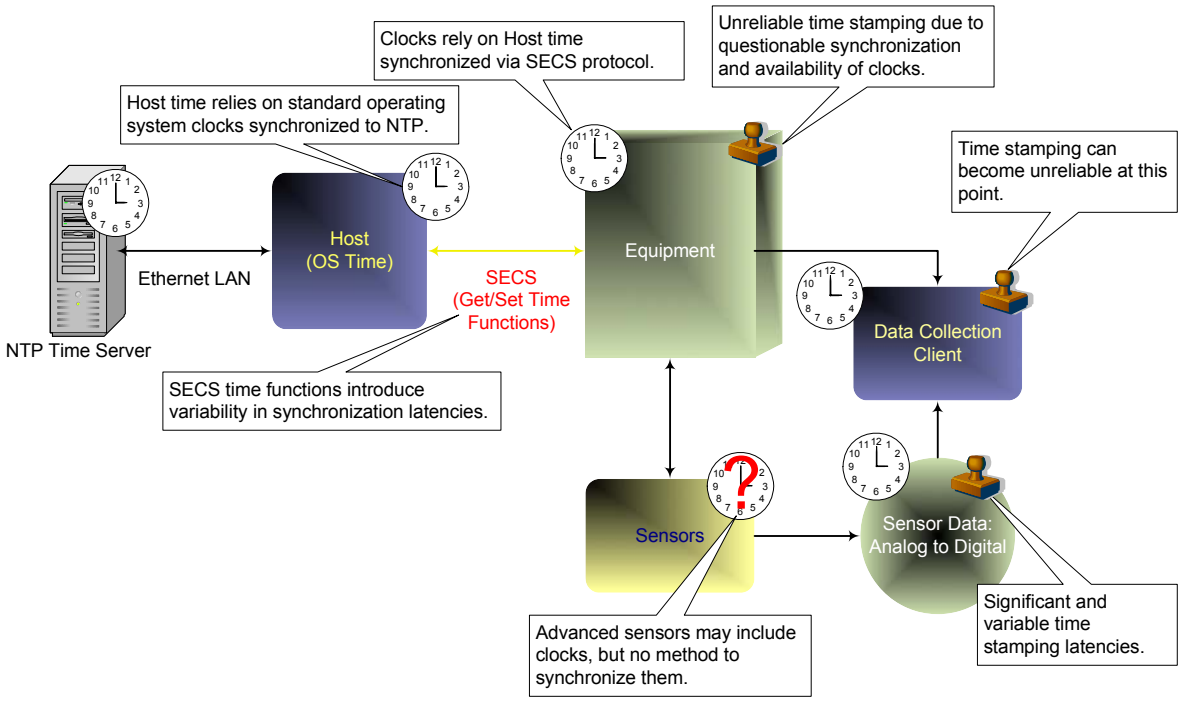


Figure 2 Degradation of timing data accuracy in a typical semiconductor factory network.

An unsynchronized tool module can introduce significant time stamping latencies. In addition to the existing latencies and variability in latencies between the host and subsystems, data can incur greater delays as they are being transferred or buffered during collection before it is time stamped.

In addition to clock communication issues, the equipment is inherently plagued with other timing challenges. Equipment processors are often overloaded with tasks, and synchronization is among the lowest of priorities. The necessary interrupts required to consistently increment time may often be overridden by higher priority processes. There are also clocks that do not offer the resolution needed for 1 ms accuracy or require frequent synchronization due to its poor quality. In industry standards, most require equipment to have only centisecond resolutions. The error handling mechanism typically aggravates the synchronization process by further overloading the CPU. However, the key obstacle to achieving synchronized time is the inability to interface with the clock modules provided by third party manufacturers. Often these black box modules have no interface to communicate the clock information.

Historically the industry has always been pushing the limits of technology without concern for synchronized time. Therefore the architecture has always been designed to ensure priority for wafer processing and lacks support for low-level functions such as ensuring synchronized timing. Because of the many factors contributing to synchronization latencies, numerous challenges exist before the network environment would be amenable to accepting a mainstream clock synchronization protocol.

Potential for 1588

Because of its promise for accurate time stamping of data and its ability to accommodate the variety of devices for industrial measurement and control requirements, the IEEE 1588 standard is a potential solution for use in semiconductor manufacturing.

Currently accuracy capability exceeds industries projected needs which will provide flexibility and time for industry to react should requirements become more stringent. Unfortunately, the current industry solution, NTP, would not be sufficient to meet more stringent requirements than the needs currently projected. In addition, because the purpose of NTP is to provide Internet-based time synchronization, the goal does not necessarily align with industrial needs where time synchronization may be more stringent, but is only required over a local area network. IEEE 1588 may contain more provisions for conserving limited computing resources.

Another benefit provided by 1588 is current adoption by industrial protocol providers such as DeviceNet. It is also able to accommodate low-level devices such as sensor and actuators as well as the high level equipment controllers. IEEE 1588 also supports both TAI and UTC time, where NTP relies on UTC. In a factory environment where global time synchronization is not always necessary, TAI may be a preferred time source as opposed to UTC.

The IEEE 1588 standard also strives towards making the 1588 standard meet industries growing needs. It achieves this by providing key features such as fault tolerance which may be beneficial for e-Manufacturing needs as more critical applications rely on synchronized time.

Challenges for 1588

The key challenges for 1588 are the cost-benefits of transitioning into a new synchronization technology as well as the factory preparedness for clock communication with a standard mainstream protocol.

Because IEEE 1588 commercial products have not yet matured, it is difficult to estimate the cost-benefits of adopting 1588. NTP comes at a relatively negligible cost, and is already a part of most factory environments. The industry has many other priorities to consider, and typically timing will fall short especially if a sufficient solution already exists. For 1588 to be adopted, the cost would have to be relatively low.

How to deploy 1588 in a factory environment with all the equipment and equipment module issues is also a challenge. Third party vendors must be reassured that exposing timing information and clocks would not compromise their intellectual property. Legacy systems should also be considered, since realistically, 1588 would not be deployed in the entire factory. Such systems would also have to co-exist and interoperate with NTP. Initial industry studies of 1588 have also indicated the devices have inherent latencies that limit the capabilities of 1588. Current devices available in semiconductor manufacturing may not be amenable to processing 1588 information rapidly enough to achieve sub-millisecond synchronization precision.

Equipment security has become an industry concern. As synchronized time will become an asset to the company, it should not be a point of vulnerability of external or internal attacks. To ensure that only a trusted master clock can adjust a particular systems' time, authentication of timing messages should be an option available in 1588.

In addition to accurately time stamping data, factory engineers need to be able to obtain data at higher sampling rates to ensure critical process events are not overlooked. Currently the data collection rates range about 10 to 100 data points per second. To justify higher synchronization requirements, we need to be able to acquire higher data sampling rates. The industry may be able to leverage the capabilities of 1588 to trigger data sampling at more consistent and higher rates.

While the need for synchronized clocks in the semiconductor manufacturing environment is burgeoning, the industry must overcome the key challenges before the factory environment can reap the benefits of 1588. While data is seen as an asset, currently data acquisition remains second priority to the process need. Therefore, the cost of migrating towards 1588 in terms of training, new software and equipment interfaces may hinder adoption of the new standard.

Potential Solutions

Some potential solutions for facilitating the adoption of 1588 include the creation of a IEEE 1588 deployments guideline/requirements document, adding security features as available in NTP, and providing open-source software implementations on a variety of platforms for trial purposes to encourage adoption.

The IEEE 1588 deployment guidelines document should be compiled by a team of experts who have deployed the solution in an industrial environment and is able to provide insight into architecting a network to provide varying precision needs. The document could describe the basic components required to deploy 1588, and the components that are optional for achieving better results. The guideline should also provide a method to verify 1588 compliance and also demonstrate how to monitor and verify synchronization accuracy of the 1588 clock nodes. The guidelines can also provide methods to acquire 1588 synchronization statistics in order to effectively monitor the overall timing performance of the system.

Security is also an important issue for ensuring the designated master clocks are intended for their role and would be fortified against vulnerabilities. 1588 should provide either a mechanism or recommendations to users in methods to prevent accidental and malicious alteration of time.

Open-source software implementations in a variety of languages and platforms, including real-time operating systems, would also be beneficial in encouraging the adoption of 1588. As many of the devices will be controlled by real-time operating systems, a software-based 1588 implementation would be of significant interest to the semiconductor industry. Even if the current industry requirements do not require the advanced precision capabilities of 1588, the flexibility for future use still makes the protocol worthwhile to explore. If the implementations are available for trial use, it will allow the industry to migrate towards 1588 as a future, more so than if the cost exceeds the current benefits.

Achieving viable clock synchronization in a semiconductor factory requires significant effort within the semiconductor industry. The industry is currently determining future activities to begin addressing their time synchronization issues and to open up equipment communications for accepting mainstream clock synchronization protocols.

As illustrated in Figure 3, the factory information control system (FICS), which is the host responsible for managing a piece of equipment, is typically synchronized to a dedicate NTP server or router over Ethernet. The host typically has a standard PC or UNIX interface, with a control client and a data client. Communications between the host and the equipment relies on an industry standard messaging protocol known as Semiconductor Equipment Communications Standard II (SECS II) [5]. Legacy communication between the host and equipment was done over RS232, but is now in the process of migrating towards an industry standard known as High-Speed SECS Messaging Service (HSMS) [6]. Newer equipment tend to use SECS II over HSMS. HSMS is a TCP/IP based protocol and can run over, but not limited to, Ethernet. For synchronizing equipment time, the industry would like to phase out get and set time functions of this protocol and allow equipment clock synchronization to be done directly with a time synchronization protocol.

The industry may also look towards requiring synchronized clocks and the ability to generate time stamps as part of their equipment model, which can be expressed in their Generic Equipment Model Standard [7]. They would also like to define a standard method to interface with module and subsystem clocks for facilitating time stamping and clock synchronization.

The Sensor Bus standard [8], allows devices such as sensor, actuator, and controllers to interoperate over DeviceNet, Profibus, Ethernet and enables rapid integration of new devices. Advanced sensors are being produced for semiconductor manufacturing to include clocks with 8-bit support. Despite the limited resources of sensors, the devices will also need to be synchronized for to support accurate time stamping features for the data it data collects.

Currently the industry has a standard architecture for integrated metrology modules to ensure rapid integration with the rest of the equipment. The standard requires synchronization of all clocks within the modules [9], and can be extended to state the exact requirements and methods for verifying accuracy compliance. A standard format for the time stamps throughout the industry will greatly facilitate synchronization and data processing.

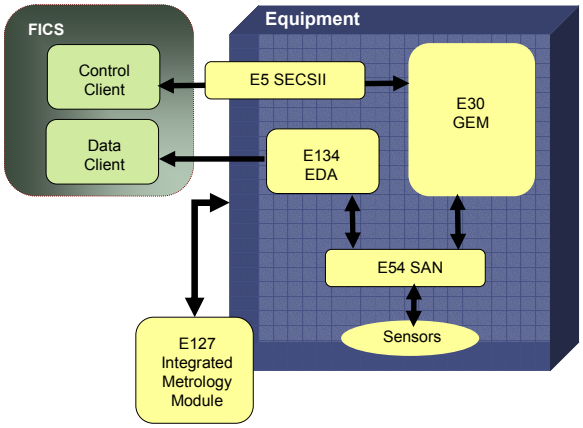


Figure 3 Industry standards needed to facilitate factory and equipment clock synchronization.

Conclusion

Realizing accurate time synchronization in a chip fabrication facility will require diverse efforts from the chip manufacturer, equipment supplier, network architects, and software vendors. Integrating a mainstream clock synchronization solution such as 1588 can be facilitated with proper guidance and continuously evolving features to meet industrial requirements.

Current semiconductor manufacturing needs for time synchronization justifies only a pure software implementation for 1588. IEEE 1588 provides several key benefits over the current Internet-based protocol, NTP. 1588 is designed to meet industrial needs by offering capabilities to interface with a wide range of devices, and offers greater flexibility by providing the potential for accuracies with hardware improvements beyond current and near-future industry requirements. Future work remains to be researched to determine whether 1588 integrates well with the devices and equipment currently in development. Testing 1588 in an actual or simulated factory environment will help determine the ease, cost, and benefits of 1588 integration. In order for 1588 to expedite adoption of the standard, the group can draft guidelines indicating basic requirements for 1588 integration and open-source implementations on a variety of platforms to ensure cost-effective testing. Additionally, security should also be examined to ensure 1588 does not introduce potential vulnerabilities into factory systems. If 1588 proves to be a cost-effective solution and continues to evolve with features intended for optimal efficiency in an industrial automation environment, it will have significant potential as a solution for synchronizing the multitudes of device clocks and contribute to the advancements of chip manufacturing automation essential to future generations of commercially viable integrated circuit technologies.

References

[1] Ghani, T., Mistry, K., Packan, P., Thompson, S., Stettler, M., Tyagi, S., and Bohr M. “Scaling Challenges and Device Design Requirements for High Performance Sub-50 nm Gate Length Planar CMOS Transitors,” Proc. Symp. VLSI Tech., Dig. Tech. Papers, pp. 174-175, June 2000. http://www.intel.com/research/silicon/ieee/sub50nmvlsitech2000.pdf

[2] Makita, M., Hirai, T., and Kodama, Y. “Robust Sensing – a Technique for Controlling a Semiconductor Device Production Process,” *2003 IEEE International Symposium on Semiconductor Manufacturing*, Sept. 30 – Oct. 2, 2003, pp. 347-349.

[3] Wohlwend, Harvey. *E-Diagnostics Guidebook, Revision 2.1*, July 12, 2005.

[4] Kumar, P.R. “Scheduling Semiconductor Manufacturing Plants,” *IEEE Control Systems Magazine*, Dec. 1994, 14(6), pp. 33-40.

[5] *SEMI Equipment Communications Standard 2 Message Content (SECS II)*, SEMI E5-1104, SEMI Standards, March 2005.

[6] *High-Speed SECS Message Services (HSMS) Generic Services*. SEMI E37-0303, SEMI Standards, March 2005.

[7] *Generic model for Communications and Control of Manufacturing Equipment (GEM)*. SEMI E30-1103, SEMI Standards, March 2005.

[8] *Standard for Sensor/Actuator Network Common Device Model*. SEMI E54.1-1000, SEMI Standards, March 2005.

[9] *Specification for Integrated Measurement Module Communications: Concepts, Behavior, and Services (IMMC)*. SEMI E127-0305, SEMI Standards, March 2005.

In Search of the Key to the Lock

Clock Synchronization Requirements in Semiconductor Manufacturing

Ya-Shian Li
John Messina

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Outline

- Industry Motivation
- Industry Application and Needs
- Distributed Synchronization Issues
- Potential Application for IEEE 1588
- Challenges in Adopting IEEE 1588
- Summary

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Motivation

Advanced Process Control

- Decreasing critical dimensions
- New material introduction
- Decreasing tolerance windows require new methods to rapidly characterize process to meet time to market and yield goals

Advanced Measurement Need

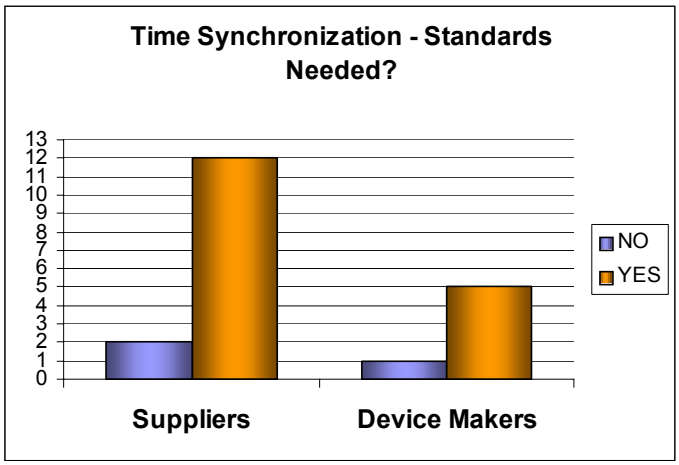
- New data interface for equipment supports up to 10 kHz data rate
- *In-situ* measurement of process condition to acquire quality data with time stamps to be able to merge data from heterogeneous sources

Other Factory Automation Capabilities

- e-Diagnostics
- Scheduling/Dispatching

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

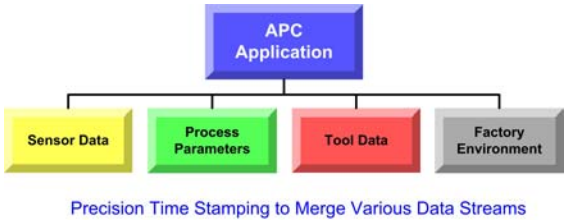
Survey: Does the industry need a standard for equipment time synchronization?



Source: ISMI/SEMI FAST III Survey, July 2005

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Advanced Process Control



Precision Time Stamping to Merge Various Data Streams

Advanced Process Control:

- Fault Detection Classification
- Process Optimization

Precision Time Stamping:

- Merging data from heterogeneous sources
- Improve multivariate, advanced correlation and analysis
- Expose new cause-effect relationships

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Advanced Measurement

Integrated Measurement:

- In-situ metrology using sensors to deliver measurement data in real-time
- Sensor data and tool data should be time stamped with synchronized clocks
- IM drives APC applications such as FDC, run-to-run, and SPC
- FDC sensor data is analyzed in real-time and/or after the wafer run to detect process failures

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

APC Example: Chemical Vapor Deposition

Chemical Vapor Deposition (CVD):

- Use of a combination of dopants (various gases) to create an ionized wafer surface with semiconductor properties.
- *In-situ* sensing and real-time control requirement driven by:
  - Restricted properties of thin films for ULSI devices
  - Increased chemical complexity

Advanced Process Control of CVD Process:

- Smaller chip sizes, decreased tolerance windows
- Effective CVD processing requires precise knowledge of the extant gas and surface chemistries during etch and deposition processes
- Environmental effects: Gas flow, temperature, pressure
- Data merged from surface, chemical gas distribution, and environmental effects could reveal additional process knowledge

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Advanced Automation

e-Diagnostics

- Remote real-time monitoring, maintenance, diagnostics, and repair of equipment on factory floor by global network of suppliers
- Accurately synchronized clocks enable rapid tracing of events for determining cause of tool fault

Scheduling/Dispatching

- In typical wafer fabrication plants there are dozens of process flows
- Each process flow can have between 200 – 500 processing steps
- Reducing equipment/wafer lot wait time requires timely estimates of completion and arrival times
- Better dispatching and scheduling rules supported by better timing synchronization for optimal factory performance

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Industry Needs

Challenges:

- Unsynchronized sensor/tool data time stamps
- Bandwidths/polling rates vary greatly
- Difficult to synchronize data streams
- Unsuitable for multivariate analysis algorithms

Application Needs:

- Current FDC requirements (10 ms to 1 ms time stamp precision)
- Data acquisition interface supports higher data sampling rates, and may require time stamp precision below 1 ms
- Clock and time stamp traceability

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Current Factory Synchronization

Existing Factory Environments:

- Comprised of heterogeneous equipment form multiple vendors
- Tied together with industrial protocols such as:
  - DeviceNet
  - Ethernet/IP
  - Profinet
- Key obstacle is clock communication within the factory equipment

Factory Networks:

- NTP using dedicated servers/routers to propagate synchronized time to factory computing systems including equipment host
- Advanced factories can support 1 ms message latencies which allows full utilization of NTP protocol

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Current Equipment Synchronization

Equipment subsystems/modules manufactured by third party

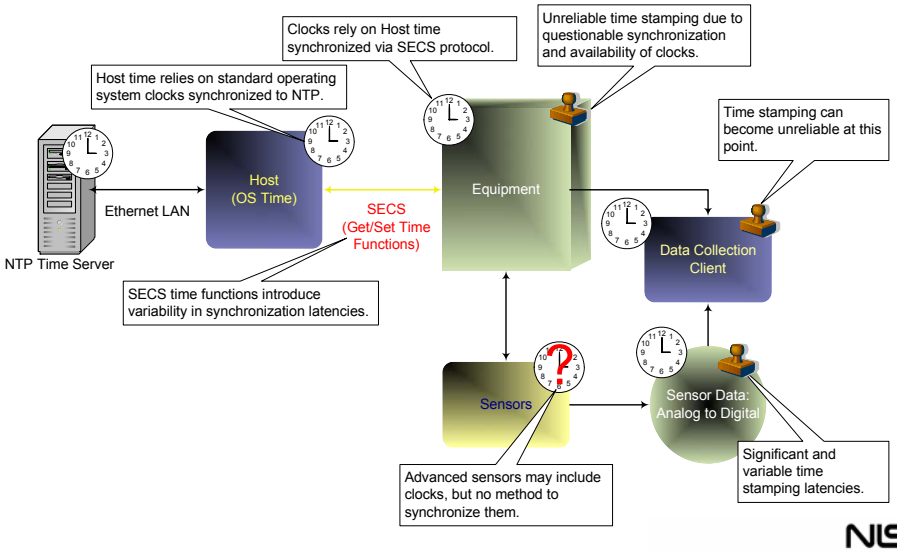
- Black box modules, inaccessible internal clocks
- Lack uniformity on time stamping formats
- Lack external synchronization controls
- Can introduce significant latencies between host and subsystems
- Clocks can vary from 100 ms to 2 minutes within a single piece of equipment, if synchronized at all

Equipment not architected to support clock synchronization

- Synchronization from equipment host to equipment through SECS/GEM messages
- Rapid, accurate synchronization to all equipment clocks not readily available

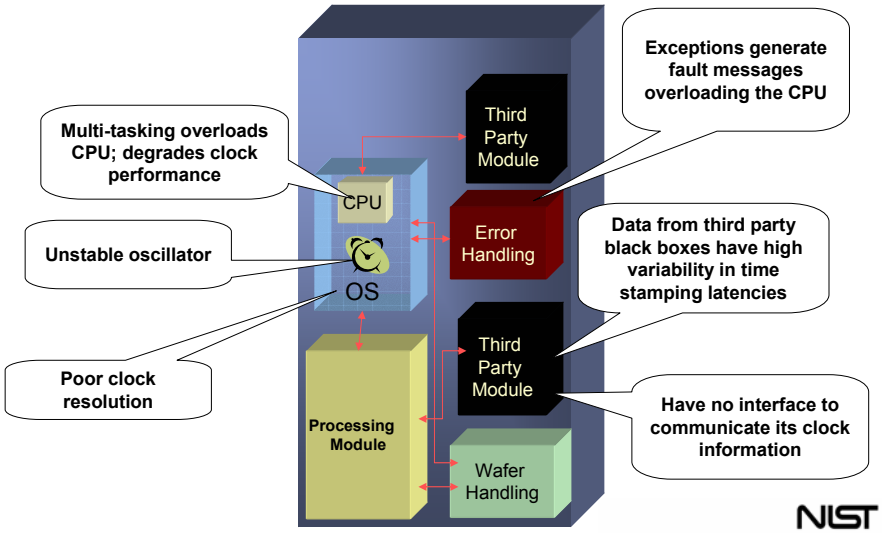
NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Issues: Factory Synchronization



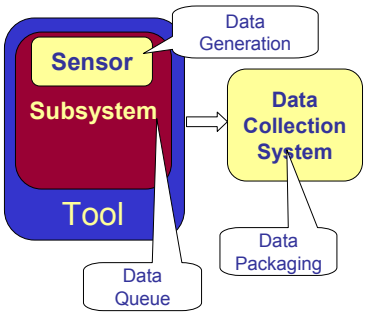
NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Issues: Equipment



NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Issues: Data Time Stamps



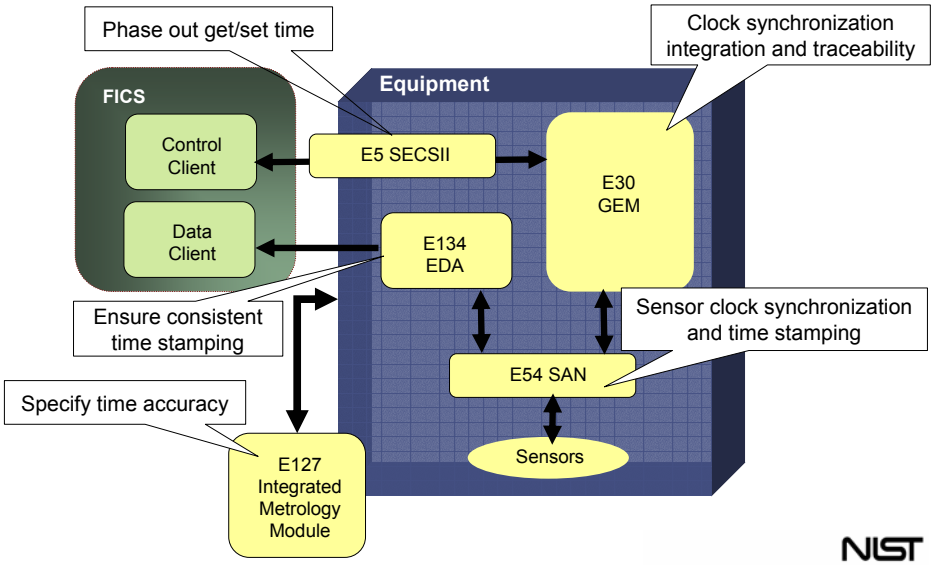
Possible Time Stamping Points

Time stamping latencies within a single tool due to inconsistencies:

- Format/time reference
- Point of time stamping
  - Should reflect time of data generation
  - Should not include:
    - Transmission time
    - Processing
    - Queuing time

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Issues: SEMI Standards



NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Potential for IEEE 1588

Synchronization Accuracy

- Meets current need, ample room for future requirements

Specialized for Measurement and Control

- Ability to accommodate a wide range of systems
- From sensors to equipment controllers
- Designed to support industrial network protocols
- Supports relative (e.g. TAI) and global (e.g. UTC) time

Evolving to Meet Industrial Needs

- Future fault tolerance capabilities may extend use of time synchronization in other areas of e-Manufacturing.

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Industry IEEE 1588 Challenges

Cost-Benefits

- Does the current need justify the cost?

Factory Deployment

- Protocol Integration
  - With current equipment architecture
  - Access to clocks without compromising IP
  - Adaptability with legacy systems

Data Availability

- Low, inconsistent data sampling rates
- Data acquisition is second priority to process need

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Future Work for IEEE 1588

Security

- Ensure master clocks are intended for their role, not compromised
- Provide mechanism/recommendations to prevent accidental/malicious change of time

IEEE 1588 Deployment Guidelines

- Factory network, equipment, clock, hardware, software etc. requirements
- Recommendations for deploying IEEE 1588 for varying accuracy needs
- Recommendations for verifying synchronization accuracy of PTP clocks

Open-source Software Implementation

- Available for industrial operating systems
- Accelerate adoption of IEEE 1588

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Summary

Time synchronization need: Accuracy of 1 ms in semiconductor equipment and factory systems for measurement and control

1 ms is the current need, although data interface capabilities support sampling frequencies up to 10,000 Hz. Accurately time-stamped process metrology and tool data will significantly improve advanced process control capabilities.

IEEE 1588 is a potential solution

IEEE 1588 provides guaranteed accuracy levels that can be extended as necessary to meet future requirements.

Challenges face factory deployment of IEEE 1588

Cost-benefits of deploying IEEE 1588 should be analyzed and justified. Integrating IEEE 1588 protocol will require adjustments to current industry standards.

IEEE 1588 recommendations needed

Recommendations for integrating IEEE 1588, while meeting cost, accuracy, and security requirements, are needed from industries requiring synchronized time in automated process control with equipment based on modular architecture.

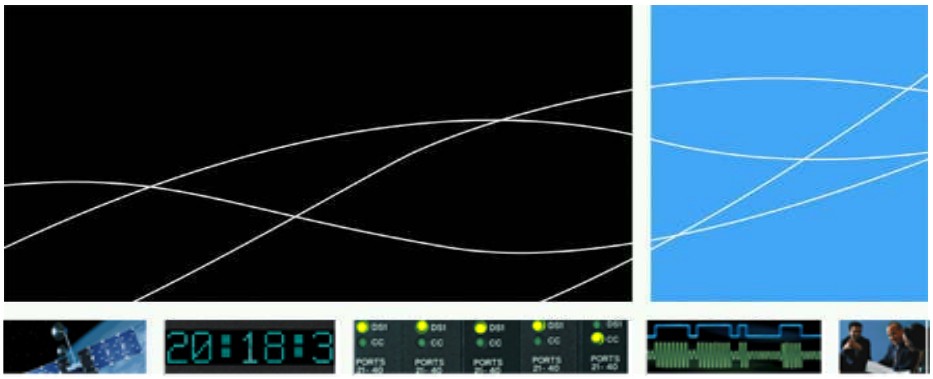
NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Questions and Comments

ya-shian.li@nist.gov
john.messina@nist.gov

Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States.

Certain commercial equipment, instruments, or materials are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.



A PTP Grandmaster Clock

Doug Arnold, Harrell Huckeba, Chris Calley, Paul Skoog
Timing Test and Measurement Division

Symmetricom

Introduction

Symmetricom

- ***What is a PTP Grandmaster Clock?***
- ***A Grand Master for Defense/Aerospace***
- ***Performance Measurements***
- ***Conclusions***

Grandmaster Properties

Symmetricom

- Time stamp accuracy to internal clock
- Time stamp accuracy to absolute time
  - Grand masters at opposite ends of network should have phase alignment better than the network allows
  - Some applications require true time of day
- Reliability: MTBF
  - Redundant subsystems
  - Oscillator holdover

Grandmaster Properties

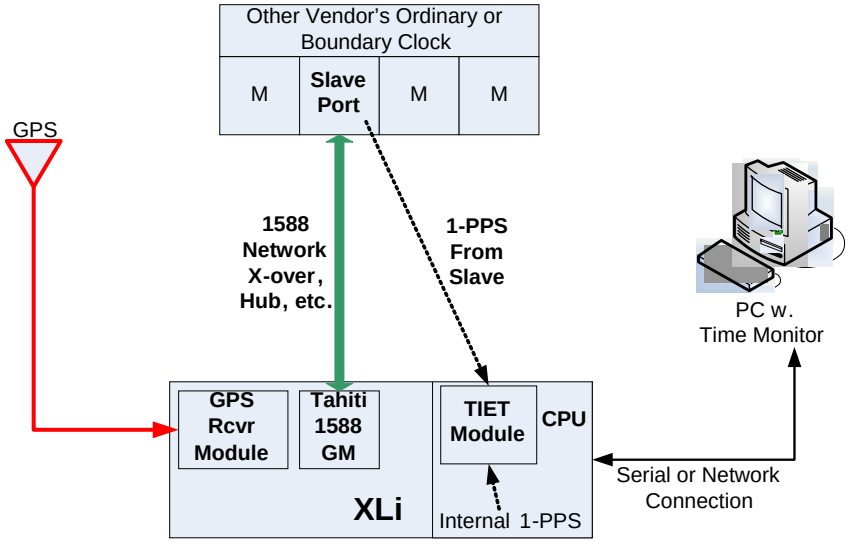
Symmetricom

- Frequency stability
  - Easier to lock to low flicker/random walk noise
  - Specified by Allan Deviation vs. averaging time
- Interoperability with legacy systems
  - Other time scales: GPS, UTC
- Non PTP functionality
  - PTP is a *feature* not a *product*

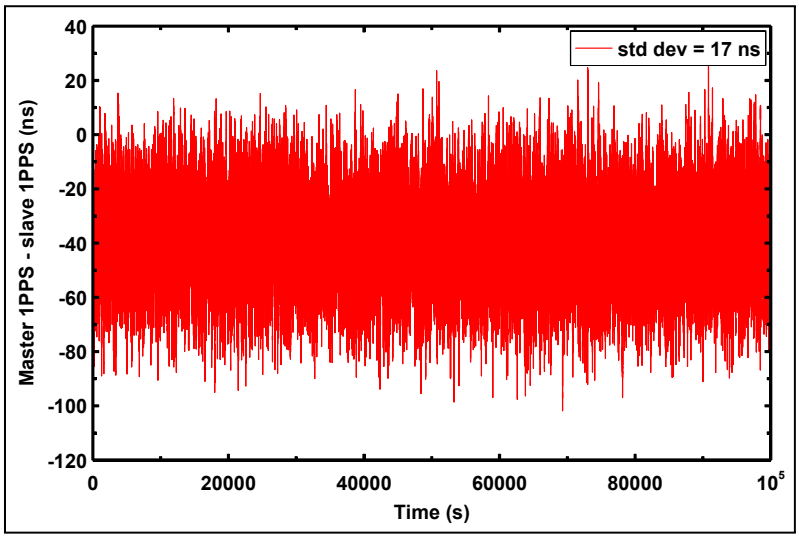
XLi IEEE 1588 Grandmaster

- Based on XLi modular time and frequency system
  - Tailored for Defense/Aerospace
  - Traditionally used serial time codes on dedicated wires
- Redundant subsystems
  - GPS receivers
  - Power supplies
  - PTP grandmaster option cards
- Many non PTP features

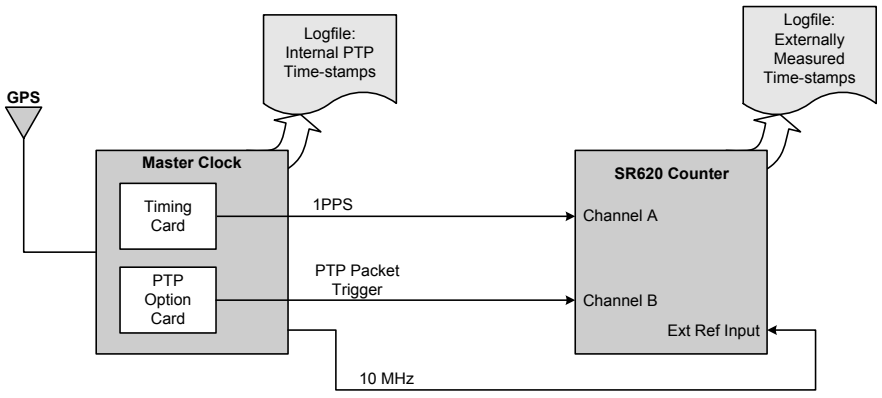
Synchronization Test Setup



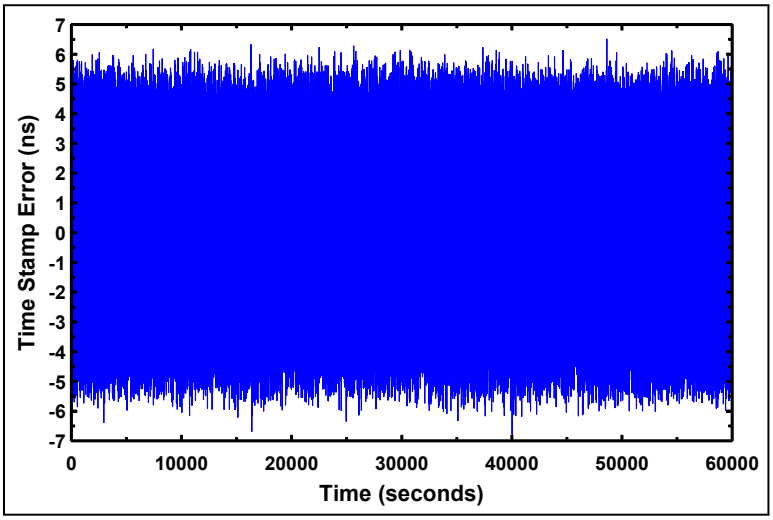
Test Configuration andResults



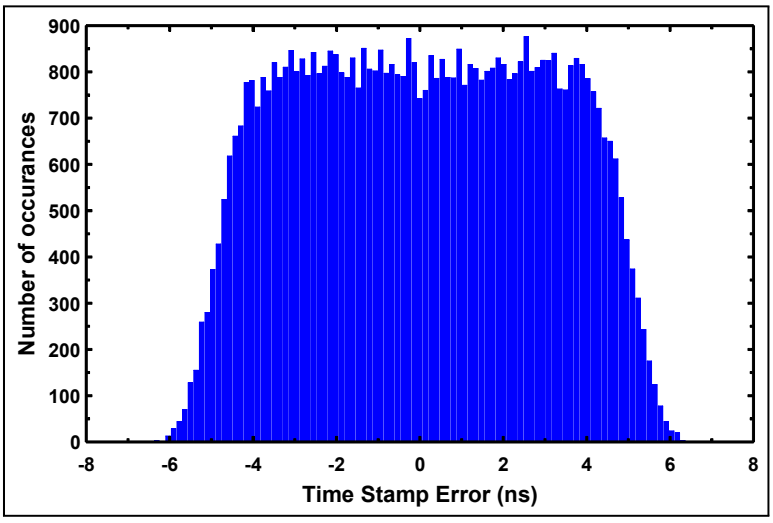
Measurement Set up: Time-stamp Accuracy



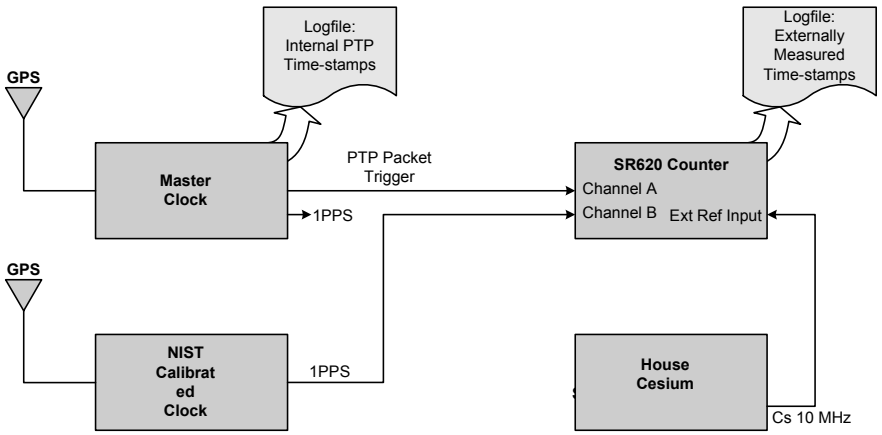
Data: Stamp Accuracy



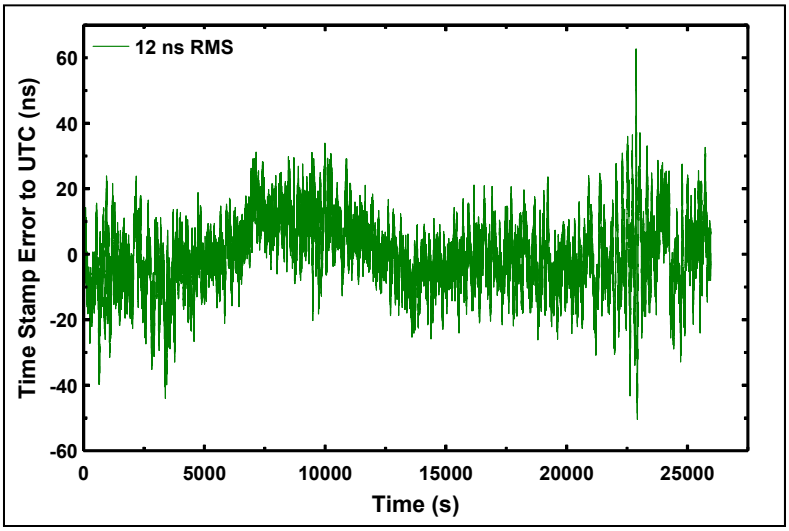
Data: Time-stamp Accuracy



Accuracy to UTC



Data: Time-stamp Accuracy



Conclusions

- IEEE 1588: a feature not a product
  - Built in measurement capability
  - Non 1588 time and frequency inputs/outputs
- Reliability
  - redundant subsystems
- Time stamp accuracy
  - Dominated by 100 MHz internal counter
  - 12 ns RMS to UTC:USNO (GPS)
  - synchronization with slave 17 ns RMS

Report on the P1588 Activities to Extend IEEE-1588-2002

John Eidson
October 11, 2005
john_eidson@agilent.com

P1588 Agenda & Status

1. **Resolution of known errors in IEEE 1588-2002:**
   - **Status: completed**
2. **Enhancements for increased resolution and accuracy:**
   - **Extension fields to allow sub-nanosecond time stamps: correction field extensions on appropriate messages for ALL corrections**
   - **Shorter sync intervals allowed**
   - **Status: General agreement, details TBD**

P1588 Agenda & Status

3. **Prevention of error accumulation in cascaded topologies:**
   - **End-to-end transparent clock (residence time corrections on current message structure)**
   - **Peer-to-peer transparent clock (residence time & path delay corrections. Uses only Sync class messages plus path delay)**
   - **Status: In progress, target completion at face-face this week**

P1588 Agenda & Status

4. **Rapid network reconfiguration:**
   - **Path delay measurement mechanism (A-Delay)**
   - **Rules for using these measurements to correct timestamps**
   - **Particularly applicable to peer-to-peer transparent clocks**
   - **Status: In progress , target completion at face-face this week**

P1588 Agenda & Status

5. Optional shorter frame:
   - Primary target is telecom and similar environments
   - Provision for unicast to manage scale issues
   - Status: In progress
6. Master clock redundancy:
   - Master-centric: (multiple masters self-select active master)
   - Slave-centric: (slaves select from several active masters)
   - Status: In progress

P1588 Agenda & Status

7. Security extensions:
   - Authentication of grandmaster
   - Reference other standards
   - Difficult, varying requirements, BC and TCs
   - Status: Initial discussions have bounded the problem
8. Ethernet layer 2 mapping:
   - Status: Initial discussions only

P1588 Agenda & Status

9. Mapping to other protocols:
   - DeviceNet
   - MPLS
   - Status: Initial discussions only
10. Annex D modifications for variable Ethernet headers:
   - Tagged frames, QOS, IPV6.
   - Status: Initial discussions

P1588 Agenda & Status

11. Conformance enhancements:
   - 1 PPS or equivalent signal,
   - Management message or extension fields to make internal time stamps visible,
   - Status: Not started
12. Increased system management capability:
   - Additional management messages,
   - Perhaps SNMP
   - Status: Not started

P1588 Agenda & Status

13.Extension mechanism:

- **Uniform way of extending fields/messages.**
- **Status: agreement in principle**

General:

1. P1588 meetings:

- **1st and 3rd Thursdays of each month**
- **Face-to-face about 3-4 times a year**
- **30-40 active members**
- **Lots of sub-committee work**
- **Open to all- see http://ieee1588.nist.gov for details**

General:

2. Target completion date for IEEE 1588, Version 2:

- **Ballot in June-Sept. 2006 time frame**
- **Publication early 2007**

Overview and Results of the 2nd IEEE1588 Plug-fest

by Dirk Mohl
Hirschmann Automation and Control GmbH

Overview

Goal

- Interoperability
- Heterogeneous Environment
- Proof of functionality
- Giving everybody the possibility to verify its implementation

Overview

13 Companies participating in the Plug- fest

Agilent Laboratories
Hirschmann Automation and Control GmbH
IXXAT Automation GmbH
KUKA Controls GmbH / KUKA Robotics Corp.
MEINBERG Funkuhren
National Instruments
Westermo OnTime AS
Resolute Networks Inc.
Rockwell Automation
Semtech
Symmetricom Inc.
VXI TECHNOLOGY
Zürich University of Applied Sciences

IEEE1588 Plug- fest 2005

The Basis

- several Test documents

What we have done

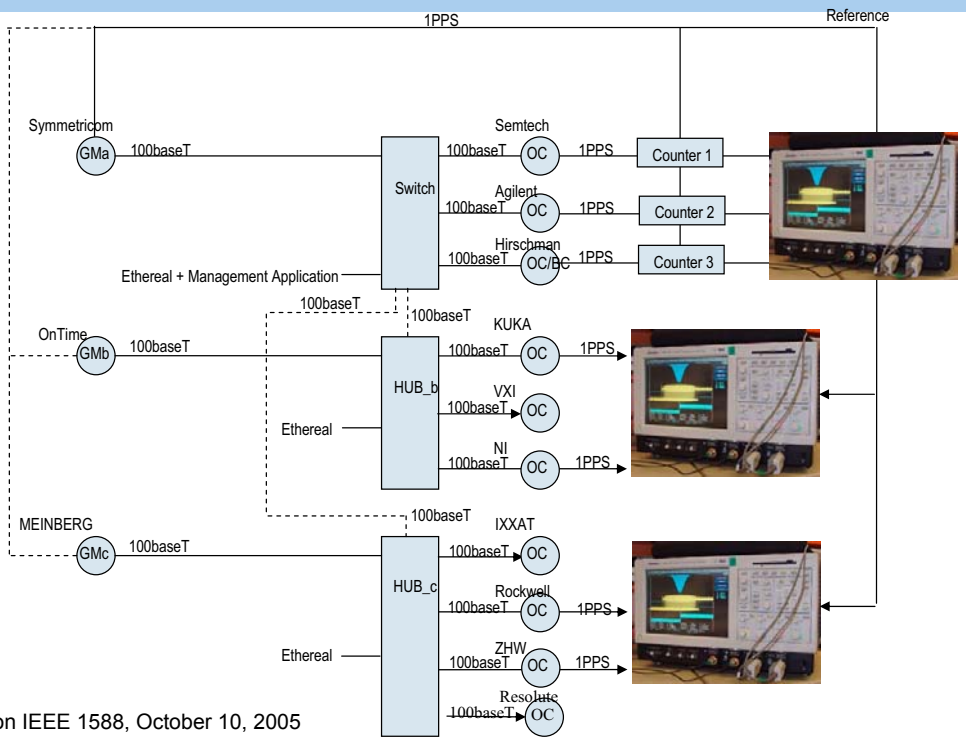
- Interoperability
- Synchronicity
- Best Master Clock Algorithm : UUID, Stratum, Variance, PM
  - GM, Switch - over and Synchronicity
  - Clients
- Management
- Load Test

Objects under Test

- 3 HW based Grand Master Clocks
- 6 HW based Ordinary Clocks
- 1 HW based Boundary Clocks
- 2 HW based Transparent Clocks
- 3 SW based Ordinary Clocks
- 1 SW based Boundary Clock

IEEE1588 Plug- fest 2005

Network Setup



IEEE1588 Plug- fest 2005

Results

- Most OC have synchronized to GM
- Synchronicity jitter between 20 to 100ns for HW Clocks
- 10 to 100µs for SW- based clocks
- Offset: GPS to PTP 0 ... 50ns
- Offset GM to PTP PPS : 0ns; 60ns; 300ns
- BMC worked in most cases as expected, GM, OCs
- Network Load:
  - HW- based Clocks no problem
- Open points: Offset calibration for GPS and PPS

  => some minor issues have been detected
  => success ! most devices work together very good

IEEE1588 Plug- fest 2005

Demo

- **You have further questions ?**

=> IEEE 1588 implementations and additional information will now be presented in the two rooms

Using PTP for Synchronizing Legacy Networks

Martin Burnicki (martin.burnicki@meinberg.de)
Heiko Gerstung (heiko.gerstung@meinberg.de)
Udo Maltzahn (udo.maltzahn@meinberg.de)

Meinberg Funkuhren
Auf der Landwehr 22
31812 Bad Pyrmont
Germany

http://www.meinberg.de

Document Version: 1.12 / October 22nd, 2005 / hg

Table of Contents:

1. Introduction 3
2. Motivation 4
3. Today's Solutions 5
4. Differences between NTP and PTP/IEEE1588 6
   4.1) Network Infrastructure 6
   4.2) Time Sources / Redundancy 6
   4.3) Accuracy 6
   4.4) Security 6
   4.5) Target Audiences 7
5. Identified Applications 8
   5.1) Application 1: Combined PTP Grandmaster Clock and NTP Time Server with GPS 9
   5.2) Application 2: Using PTP to synchronize a „slave" Time Server 11

1. Introduction

This document summarizes the contents of the presentation at the IEEE 1588 conference 2005 in Winterthur, Switzerland and tries to reflect the contents of the presentation as complete as possible.

It contains some background information and describes the approach of two PTP/IEEE1588 applications Meinberg developed as a reaction to customer demands.

The authors would like to thank the hosts of the IEEE 1588 conference and plug-fest, namely the IEEE, NIST and the Zurich University of Applied Sciences (ZHW). It has been a very interesting and valuable experience for us and we congratulate the organization team for making this an outstanding event.

If you have any further questions regarding NTP, PTP or the Meinberg product range, please do not hesitate to contact the authors or info@meinberg.de.

2. Motivation

The Meinberg customer base is mostly interested in Network Time Protocol based synchronization, but recently a few customers started to show interest in PTP/IEEE 1588 compatible products. Most of those customers need to use both protocols, NTP and PTP, to synchronize their NTP/SNTP clients and their PTP clients. Therefore using a single time source for these two worlds seems to be a cost-effective way of meeting the demands of the end users.
A growing number of companies feel the need for time synchronization in their networks. As technical advancement goes on, growing demands for accuracy let customers ask for sub-millisecond performance, which cannot be guaranteed for NTP infrastructures.

The available solutions for high accuracy time synchronization are mainly based on the deployment of multiple hardware reference clocks in all systems with such a requirement for accurate time stamping data or time controlled processes. The possibility to use standard cabling systems for high-precision time synchronization instead of installing dedicated time sync cables (e.g. for IRIG signal distribution) is an interesting and cost saving approach for most end users.

For special applications like underground installations or other locations where the reception of a time signal like GPS or long-wave radio (e.g. the German DCF77 or the UK MSF signal) is simply not possible, the alternative of using a legacy Ethernet/IP based network for time synchronization is an interesting way of getting the time signals to the point where they are needed.

3. Today's Solutions

3.1) Network Time Synchronization

When it comes to time synchronization in legacy networks, the vast majority of systems go with NTP (Network Time Protocol) or SNTP (Simple Network Time Protocol), both sharing the same packet format. They mainly differ in how a client deals with the time synchronization information it receives from the server. While NTP clients calculate the current time with a weighted average over a set of servers and smoothly adjust their clock, SNTP clients often simply take the time received from one server and step their clock to be in sync.

While NTP is reported to achieve accuracy at the microsecond level, this is possible only in ideal environments where the NTP clients and servers are facing low workloads and no asymmetric network delays are experienced. The typical performance level of NTP time synchronization is in the 1 – 10 milliseconds range, which is sufficient for most of today's applications … but not for all.

3.2) Single Node Synchronization

Where the accuracy of NTP/SNTP is not sufficient enough or where no network is available at all (i.e. in standalone systems), the common way of getting time synchronization is to integrate some kind of hardware time reference into a system.

A number of different references can be chosen from, their individual availability is depending on the geographical location and the existence of other time sources.

A short and incomplete list could look like this:

- GPS based radio clocks (antenna location needs to have a free view to the sky, globally available)
- Long wave Radio Signals (regionally different, not available everywhere) Examples: DCF77 (Germany), MSF (UK), WWVB (USA), JJY (Japan). See http://www.npl.co.uk/time/time_trans.html for a comprehensive overview
- Frequency Standards / High Quality Oscillators (setting the start time manually and feed a PPS/10Mhz signal into the to-be-synchronized system which is generated by a highly accurate frequency standard, e.g. a cesium based system, or a high quality OCXO based system)
- IRIG (needs a dedicated cable and another time source capable of generating the IRIG signals)

4. Differences between NTP and PTP/IEEE1588

4.1) Network Infrastructure

While NTP was designed for operating in legacy networks including WAN links and Internet connections, PTP's favored network environments are small and dedicated networks used for specific applications, e.g. process automation networks.

NTP is mostly used in unicast mode, i.e. a client directly requests time synchronization from one or more servers and receives an appropriate reply. A roundtrip delay is then calculated by the client based on four timestamps (T1: Client sending the request, T2: Server receiving the request, T3: Server sending the reply, T4: Client receiving the reply). In environments where reduced network traffic is preferred and lower accuracy can be accepted, NTP can be used in broadcast or multicast mode where the server periodically sends out time synchronization packets as broadcast or multicast packets. Only at startup a client sends requests in order to measure the roundtrip delay, which is then applied to all incoming broadcast/multicast packets.

PTP is a multicast based protocol, where the Grandmaster Clock (this would be a "server" in NTP terminology) sends out its sync messages as multicast packets.

4.2) Time Sources / Redundancy

An NTP client is capable of receiving time from multiple upstream servers and calculates a weighted average based on measured quality (e.g. roundtrip delay, jitter, offset) and advertised information (e.g. stratum level) of each server. When one of the servers selected for synchronization fails or does not respond anymore, the client automatically removes it from its list of selected time sources. As soon as it comes back, it will be added (given the quality factors are sufficient).

The PTP approach is to select exactly one source of time (the Grandmaster clock) with a defined algorithm (the Best Master Clock selection algorithm) and all clients are ultimatively following this Grandmaster clock. The selection of the Best Master Clock is based on a comparison of clock descriptors, i.e. the "advertised" properties of each available clock. Redundancy is currently not implemented but a subcommittee is working on adding redundancy mechanisms to the standard.

4.3) Accuracy

Where NTP reaches a level of accuracy in the microseconds, PTP is capable of performing in the nanosecond range. The typical NTP accuracy within LAN infrastructures can be between 1 and 10 milliseconds. Over WAN/Internet connections it is usually better than 100 milliseconds.

PTP performance in LAN environments has been proven to reach a level of <10 nanoseconds with hardware implementations (hardware time stamping at the MAC/PHY level), pure software implementations can be in the microseconds range.

4.4) Security

Because NTP is used in legacy network structures and even over insecure Internet connections, security mechanisms have been introduced in the protocol at a very early stage. The current version 4 of NTP includes both symmetric and public key

cryptography for signing NTP packets. The so-called autokey mechanism is used to exchange public keys without manual intervention. Because PTP/IEEE1588 was designed for separated and dedicated networks, no security features have been built into the first version of the standard. There are ongoing efforts to introduce security features in PTP, but this is still a work in progress.

4.5) Target Audiences

The two protocols were initially designed to cope with completely different demands. Where NTP is mainly used to synchronize legacy computer networks, PTP has been designed to deal with higher demands on accuracy in specific LAN-only environments, customers can be found e.g. in the process automation industry.

As other fields and markets start to demand a higher accuracy, more and more possible applications for IEEE1588 appear on the scene, all of them more or less requiring additions to the protocol standards. One example is the telecommunication industry, which is very interested in getting time synchronization over their packet networks when upgrading/changing their network infrastructures. In this field, security and redundancy are important factors and the next version of the IEEE1588 standard will have to address this.

5. Identified Applications

Two different applications were identified which could be of interest for operators of legacy networks. The first application is targeted at end users running NTP clients as well as PTP clients, the second application is using PTP to improve accuracy in a typical NTP infrastructure.

Both applications are based on an NTP Time Server with an internal GPS radio clock providing high accuracy time data received from the Satellites of the Global Positioning System (GPS), operated by the US Department of Defense. More info on GPS can be acquired here: http://tycho.usno.navy.mil/gpsinfo.html

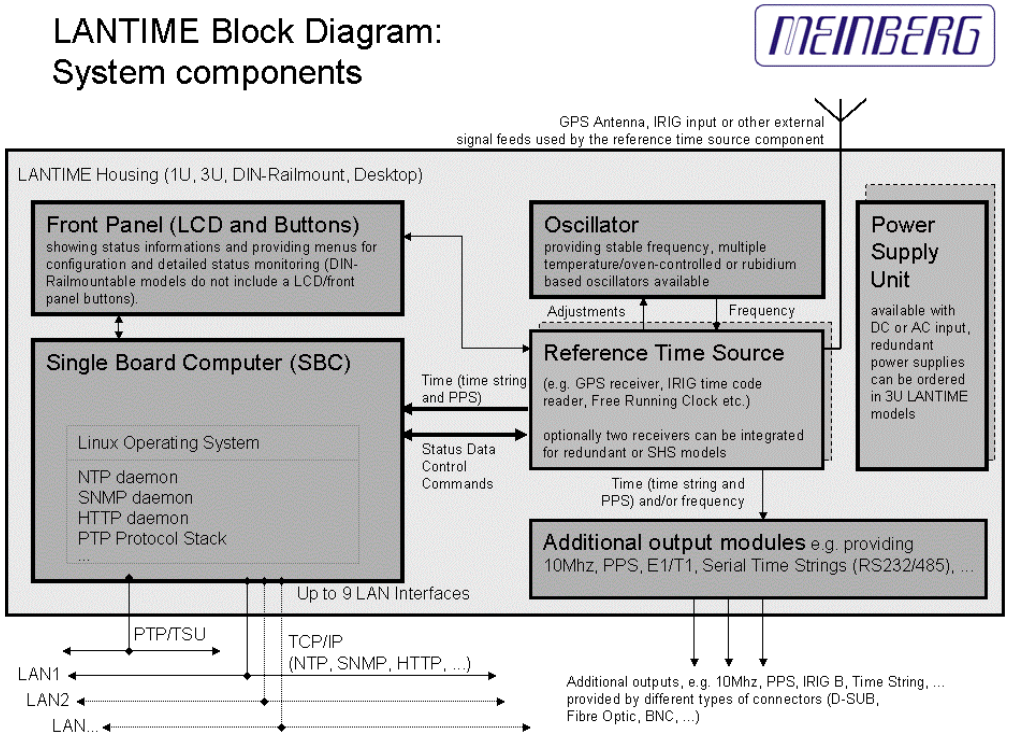
While GPS is used as a reference source for accurate time, any other time reference could be implemented to receive the current (absolute) time from any source that provides sufficient accuracy.

5.1) Application 1: Combined PTP Grandmaster Clock and NTP Time Server with GPS

In environments where both protocols are needed, a combined source of absolute and accurate time can offer several benefits:

- Both NTP and PTP client groups use the same time source (comparable time stamps)
- Equipment diversity can be reduced by integrating two related functions in one system, this means reducing costs for installation and maintenance

In order to be able to serve both NTP and PTP clients, an existing product design has been enhanced by adding an IEEE1588 compatible network port with an attached time stamping unit.



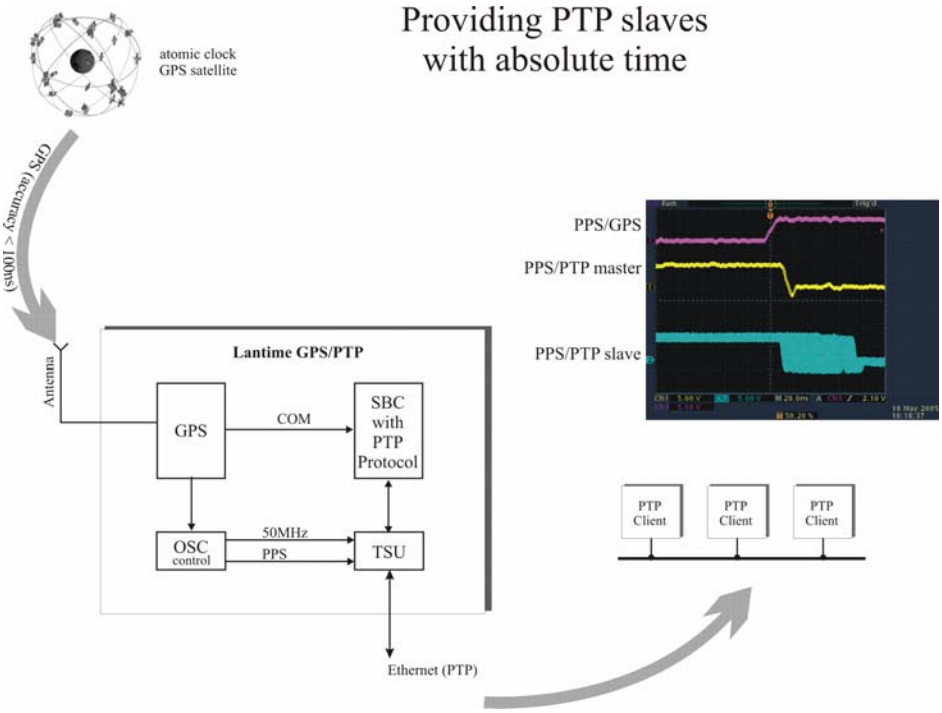
The basic system architecture of a LANTIME NTP Time Server consists mainly of a Single Board Computer (SBC), a high quality oscillator and a Reference Time Source, which is a GPS receiver in this project. After it has synchronized to the GPS signal, the receiver disciplines ("adjusts") the oscillator. This way the frequency provided by the oscillator is kept closely to the GPS frequency, which is important for having a good starting point as soon as synchronization is lost. When the GPS signal is unavailable or the receiver looses synchronization due to any other reason, it will start free running using the oscillator until GPS synchronization can be reestablished.

The NTP subsystem is reading the reference time from a serial time string and a PPS signal, both delivered by the GPS receiver.

The design of the PTP/IEEE1588 part of this solution can be described like this:

- The 50 MHz clock of the Time Stamping Unit (TSU) is derived from the high quality oscillator (OSC)
- At start-up the clock of the TSU is set to the GPS time (absolute time)
- The TSU has an additional input with which the nanosecond part can be zeroed. The PPS output of the GPS receiver is connected to this pin and is used to reset the ns part once the internal oscillator of the GPS receiver has warmed up (meaning necessary adjustment steps are under a certain limit)
- By using this mechanism, the TSU timestamp offset from the GPS PPS can be kept smaller than 20 ns

A short diagram showing the general concept of the GPS based Grandmaster clock:



5.2) Application 2: Using PTP to synchronize a „slave“ Time Server

In order to improve accuracy in pure NTP driven synchronization networks, PTP/IEEE1588 can be applied to bridge certain parts of the network maintaining a level of accuracy that could not be kept with NTP.

As a second application a LANTIME NTP Time Server was modified to additionally act as a PTP slave, using the IEEE1588 synchronization as the primary reference source for its NTP subsystem. The combined PTP Grandmaster clock/NTP Time Server with its GPS receiver (as described in Application 1) is used as the primary reference.

The benefits of this application are:

- NTP clients who are located at the other end of the network can be synchronized with a better accuracy, if a nearby NTP time server is synchronized with PTP
- Multiple output signals can be provided at the PTP slave location by using the internal high quality oscillator and discipline it based on PTP/IEEE1588 time synchronization, e.g. 10 MHz, PPS, E1/T1, IRIG B, serial time strings. This way equipment can be synchronized over the network that has no own capabilities for network time synchronization.

The basic system architecture of this solution is, identical to application 1, derived from a standard LANTIME NTP Time Server. In this case the internal GPS receiver has been replaced by a free running hardware clock and a high quality oscillator that is disciplined by using the PTP synchronization. All required frequencies and other signals can be provided based on the oscillator.

The specific software design looks like this:

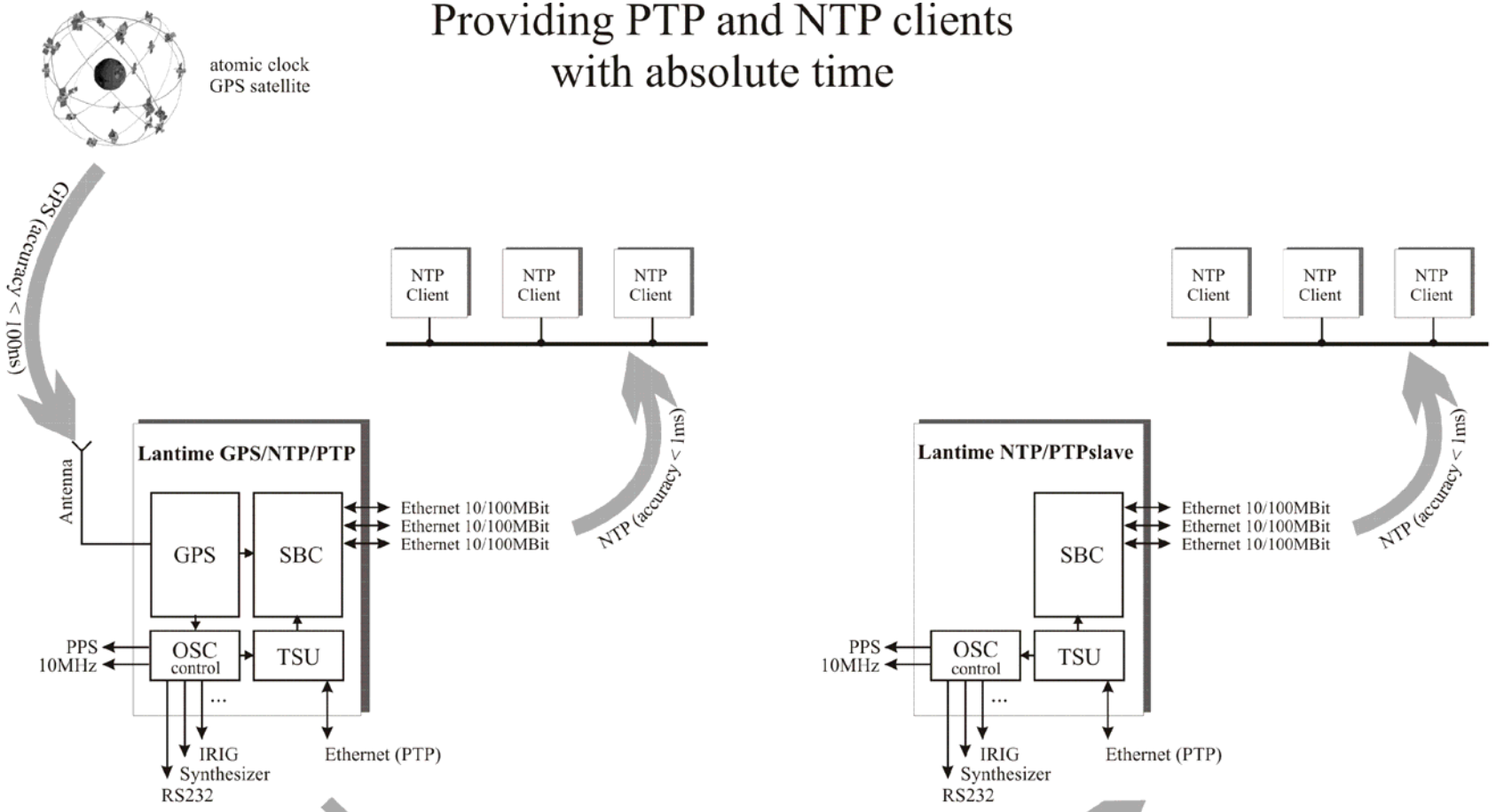
- At start-up the system waits until it is synchronized by PTP (PTP_SLAVE state has been reached)
- Now the system clock is set to the time of the TSU
- Afterwards, NTP is started. The NTP daemon has been modified with an additional refclock driver
- NTP reads the TSU‘s time by using a kernel device driver
- The TSU looks like a hardware reference clock to the NTP daemon

The NTP concept of “refclock” (=reference clock) drivers is used to feed PTP synchronized time into NTP. Per definition a reference clock has always a stratum of 0, which indicates that it is the most accurate source of time available.

For an SNTP or NTP client there is no difference in communication and handling, it simply sees that its NTP server is synchronized to a reference called “PTP” and that is has a stratum of 1 (which basically says that the server itself is getting the time from a stratum 0 source – the PTP/IEEE1588 Grandmaster clock).

In order to minimize accuracy degradation between the two LANTIME systems (the GMC and the PTP Slave system), all IEEE1588 measures like transparent or boundary clocks can be used on that part of the network.

Providing PTP and NTP clients with absolute time



Satellite controlled NTP/PTP time server
- receives absolute time from GPS
- synchronizes NTP clients
- acts as PTP grandmaster
- additional outputs (RS232, IRIG, 2048MHz ...)

PTP controlled NTP time server
- receives absolute time from PTP master
- synchronizes NTP clients
- additional outputs (RS232, IRIG, 2048MHz ...)

IEEE 1588 Conference and Plug-fest
Winterthur, October 10-12, 2005

Using PTP for Synchronizing Legacy Networks

Heiko Gerstung, Martin Burnicki, Udo Maltzahn
Meinberg Radio Clocks

- Motivation
- Todays Solutions
- Differences between PTP and NTP
- Solutions
  - Combined PTP GMC/NTP Time Server with GPS
  - Using PTP to synchronize a “slave” Time Server with a GPS based Time Server
- Comparing NTP and PTP Clients

Motivation:

- Time Synchronization over Network Links with an accuracy of <1ms
- Use an already existing stable time source for PTP networks
- Provide a high-precision time using standard network infrastructure (standard cable systems)
- Maximize the distance between GPS antenna location and time server location (e.g. underground installations)

Network Time Synchronization:

- Network Time Protocol (NTP)
- Simple Network Time Protocol (SNTP)

MEINBERG

Single Node Synchronization:
using dedicated hardware references

- GPS
- Longwave Radio
- Frequency Standards
- High Quality Oscillators
- IRIG

MEINBERG

- **Network Infrastructure**
  NTP: LAN/WAN,Unicast,Broadcast,Multicast
  PTP: LAN,Multicast
- **Time Sources**
  NTP: multiple servers, weighted average
  PTP: One single GMC at a time
- **Accuracy**
  NTP: under 1ms possible, 1-10ms typical in LAN, <100ms over the Internet
  PTP: Sub-Microsecond
- **Security**
  NTP: Symmetric Keys, Public Keys
  PTP: Work In Progress

Different Target Audiences:

PTP

- Applications with high demands on accuracy
- Relative synchronization
- Dedicated Networks

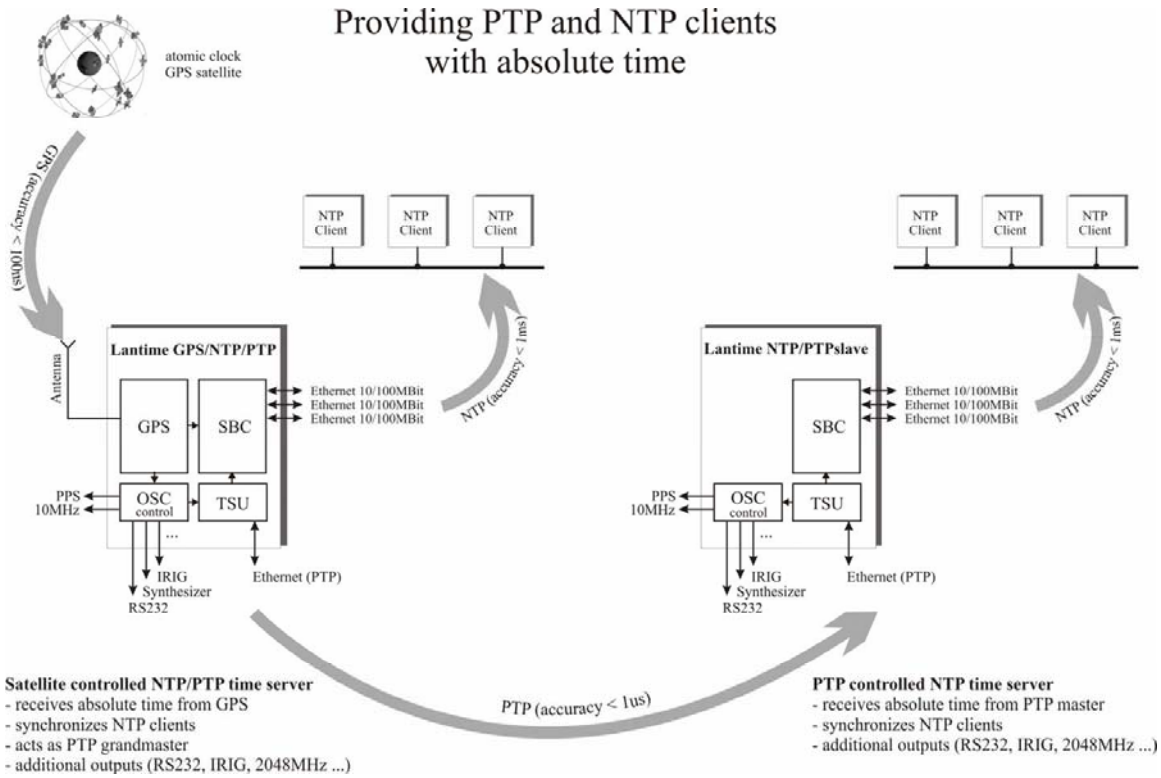
NTP

- Applications with low/medium demands on accuracy
- Synchronization using absolute time (UTC)
- Corporate LAN/WAN and Internet

Applications:

- **Time Server talking both PTP and NTP**
  → both NTP and PTP clients use the same time source
  → reducing equipment

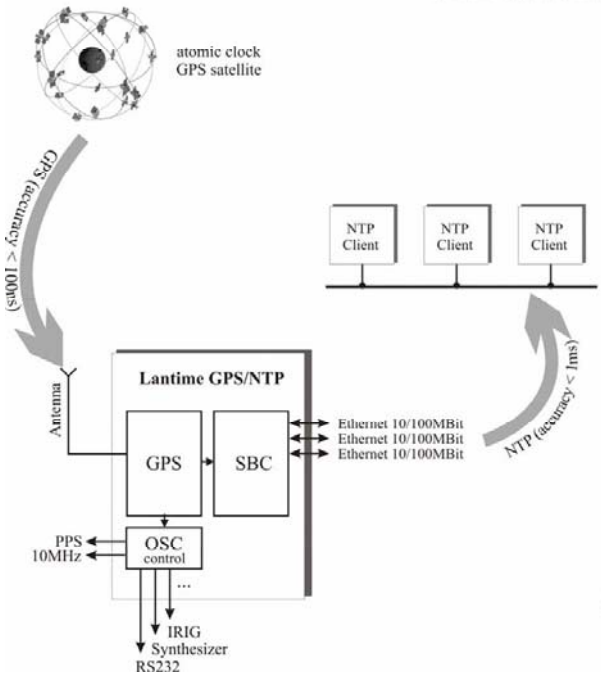
- **Using PTP for synchronizing NTP servers**
  → using PTP as a „time backbone“ for NTP infrastructures
  → reduce network related time error by using PTP instead of NTP for synchronization between two NTP servers

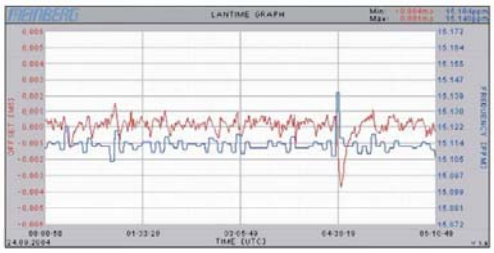


Time Server talking both PTP and NTP

- both NTP and PTP clients use the same time source, e.g. GPS, IRIG, Long Wave radio…
- all NTP and PTP clients are using comparable time stamps
- Eliminates the need to purchase, install, configure and maintain two different systems for each client community

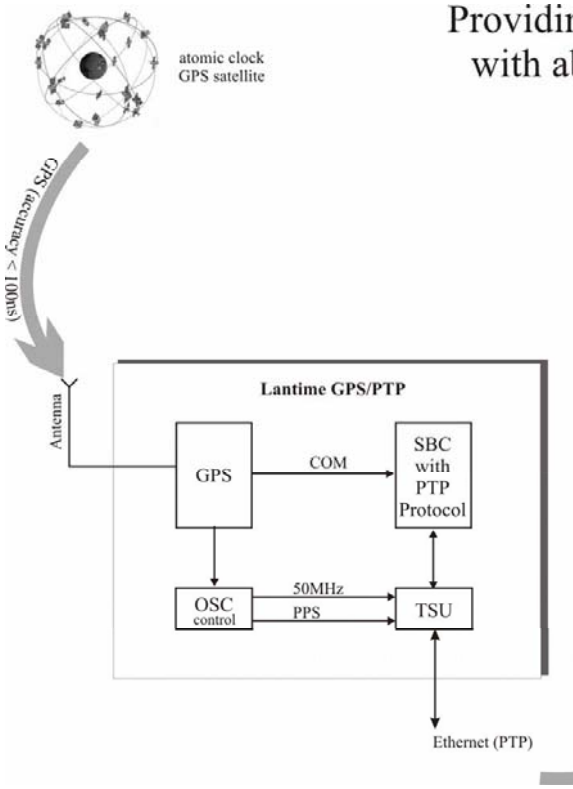
Providing NTP clients with absolute time

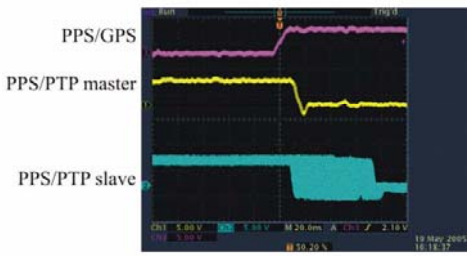


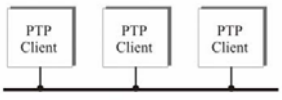


internal NTP - System time offset

Providing PTP slaves with absolute time

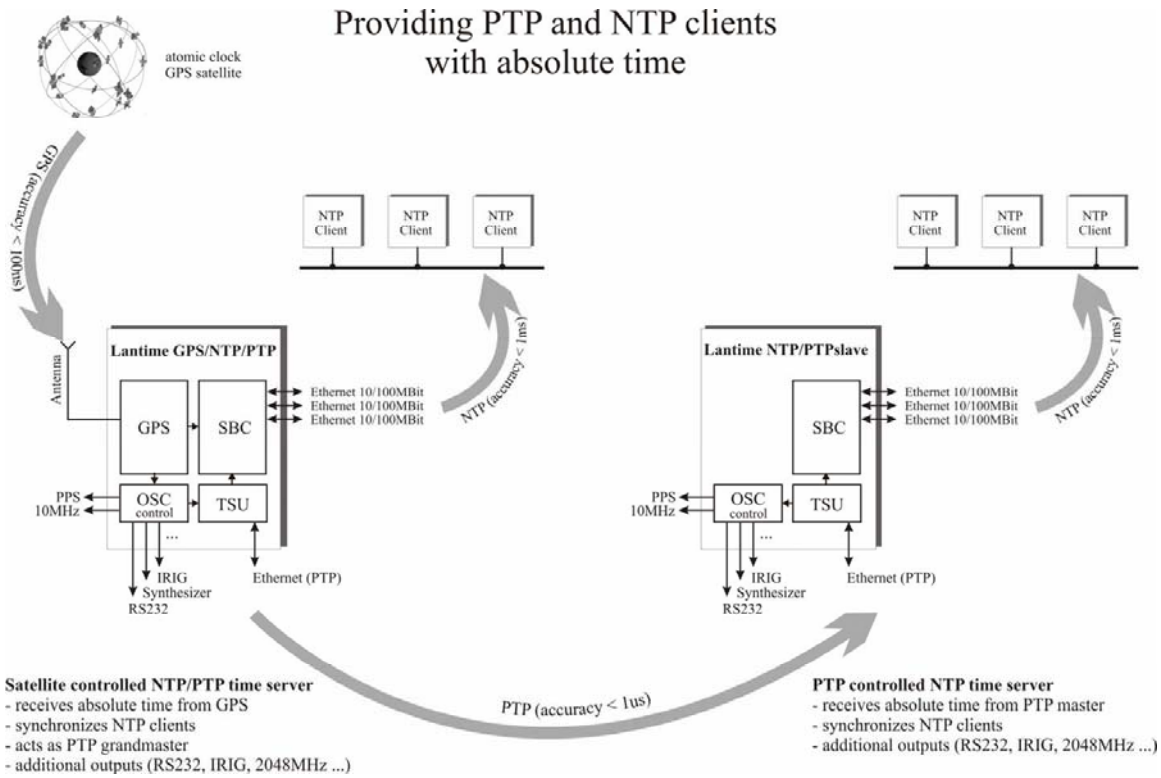






Design Overview:

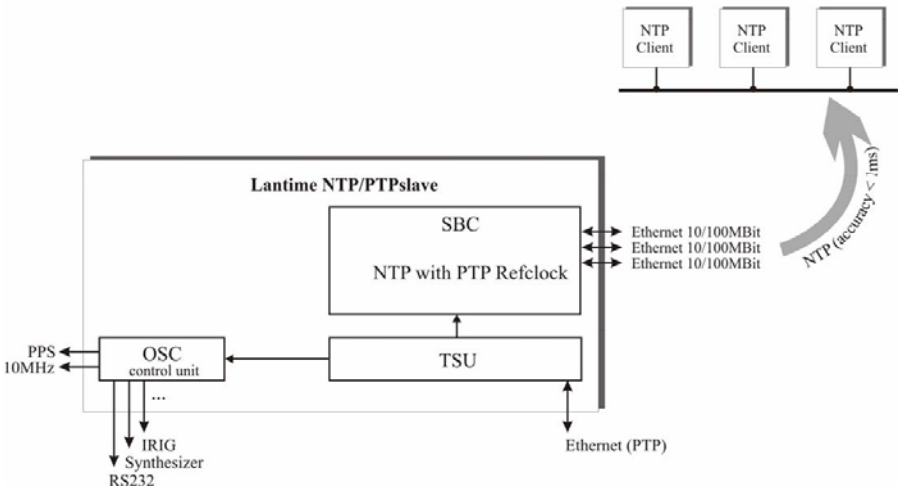
- The 50 Mhz clock of the TSU is derived from the internal Oscillator of the GPS receiver
- This oscillator is constantly disciplined using the GPS signal
- At start-up the absolute time of the TSU is set to the GPS time
- The TSU has an additional input with which the nanosecond part can be zeroed. The PPS output of the GPS receiver is connected to this pin and is used to reset the ns part once the internal oscillator of the GPS receiver has warmed up
- By using this mechanism, the timestamp offset from the GPS PPS is <20 ns



Using PTP for Synchronizing NTP Servers

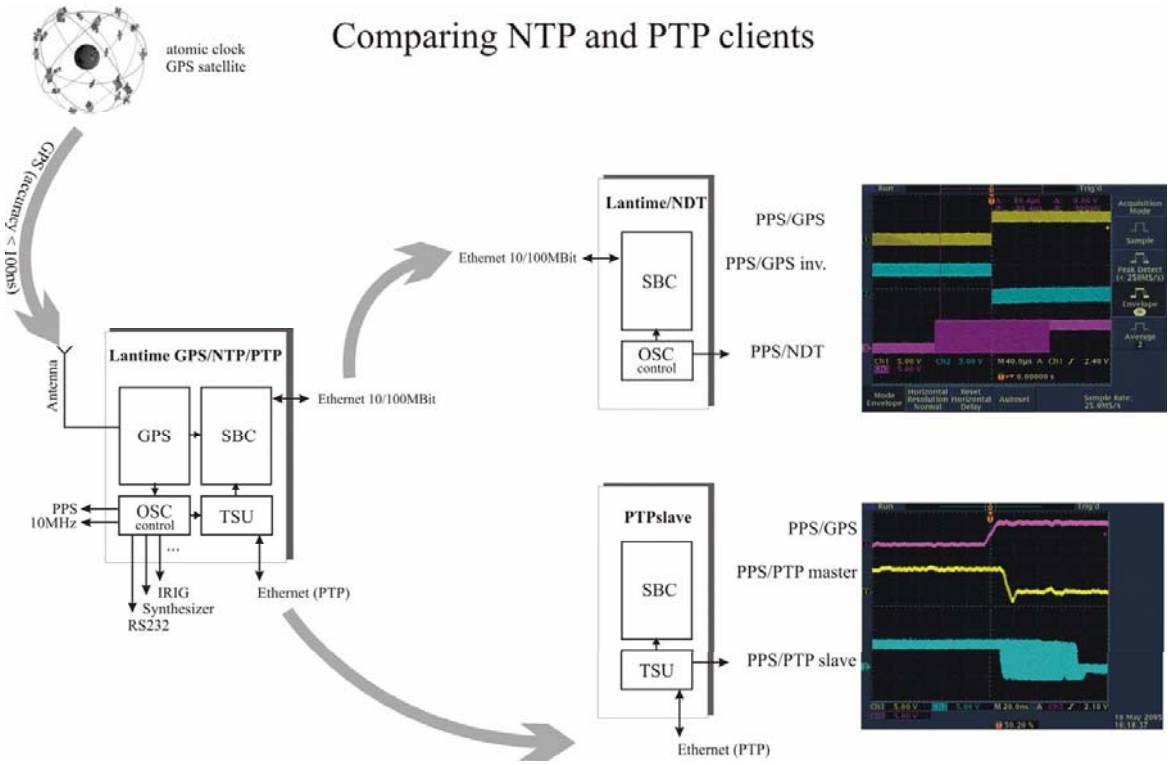
- Combined PTP GMC/NTP servers use PTP to achieve a highly accurate inter-server synchronization („time backbone“) and NTP to provide synchronization to „leaf nodes“ (which are only compatible with SNTP/NTP)
- Provide special purpose outputs (10Mhz, PPS, IRIG, E1/T1) without the need to install GPS receivers

NTP server with PTP refclock



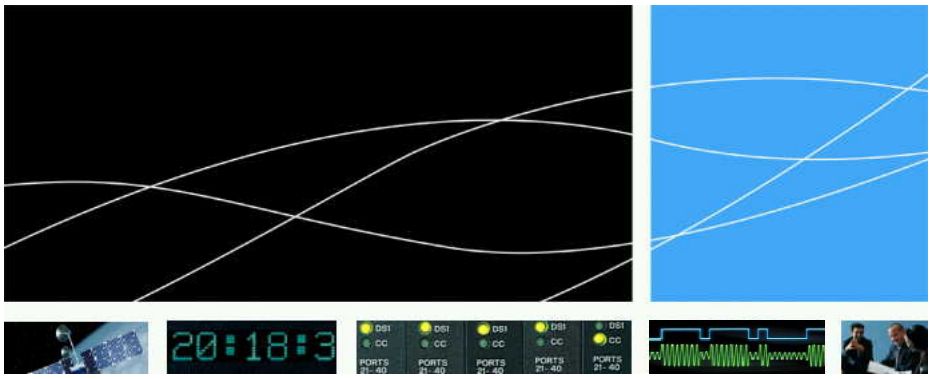
Design Overview:

- At start-up the system waits until it is synchronized by PTP (PTP_SLAVE state has been reached)
- Now the system clock is set to the time of the TSU
- Afterwards, NTP is started. The NTP daemon has been modified with an additional refclock driver
- NTP reads the TSU‘s time by using a kernel device driver
- The TSU looks like a hardware reference clock to the NTP daemon



Questions?

heiko.gerstung@meinberg.de
http://www.meinberg.de



Implementation Considerations for IEEE-1588 in Telecom Applications

George Zampetti
Vandana Upadhyay
Srini Bangalore

Symmetricom

Agenda

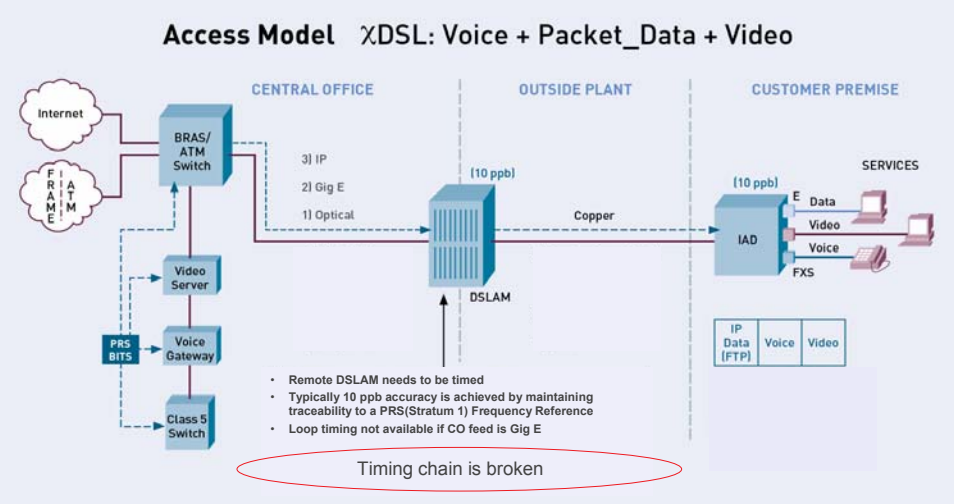
Symmetricom

- Next Gen Telecom Network Access Architectures
- Telecom Synchronization Standards
- Simulation Results
- Implementation Considerations

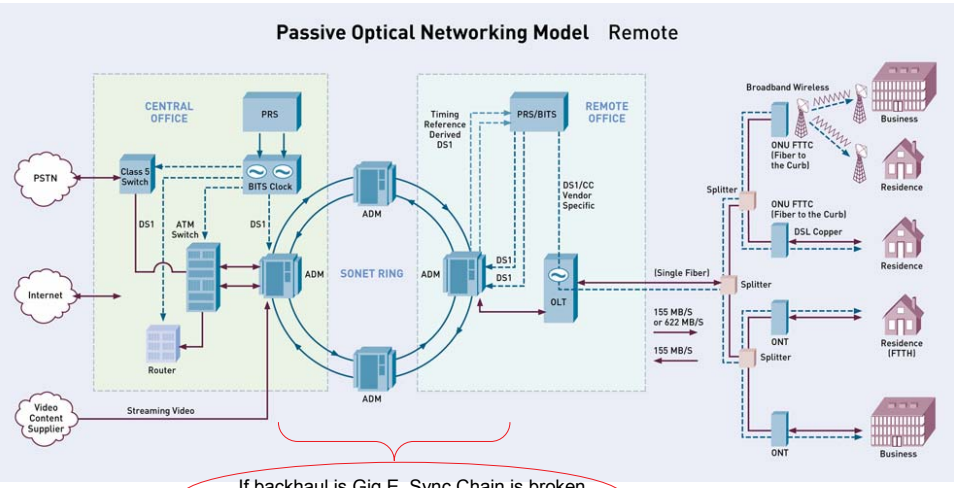
Telecom Market Drivers

- Currently NE and CPE equipment synchronization is based on direct BITS feed or using the E1/DS1 recovered line clock
- Discontinuity in sync distribution due to replacement of SONET/SDH transport with asynchronous Ethernet
- Sync distribution mechanisms such as SONET/SDH loop timing are not available in an Ethernet network
  - IEEE, IETF, ITU and ATIS are driving new standards
  - Many standards are in development - IEEE 1588, Synchronous Ethernet
- New, time-sensitive real-time applications over packet networks are emerging

Triple Play Access Architecture – xDSL



Triple Play Access Architecture – FTTH



If backhaul is Gig E, Sync Chain is broken
Local Sync Source at OLT required

Telecom Synchronization Standards

- Telecom networks must comply with several stringent synchronization standards
  - North American Industry Standards
    - GR-1244 Definition of a Clock
    - GR- 2830 Definition of a Primary Reference, PRS
    - GR- 378 Definition of an intermediate clock, TSG
    - GR- 253 SONET
  - ANSI Standards
    - T1-101 Definition of a Clock
  - ITU Standards
    - G.811 Definition of a Primary Reference, PRC
    - G.812 Definition of a Clock
    - G.823 Interface Definitions (2048 kbit/s hierarchy)
    - G.824 Interface Definitions (1544 kbit/s hierarchy)

1588 sync over Ethernet networks that bridge traditional telecom networks must meet this level of frequency synchronization

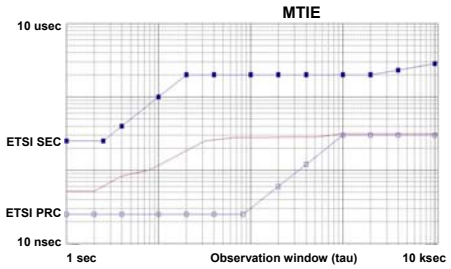
Telecom Requirements

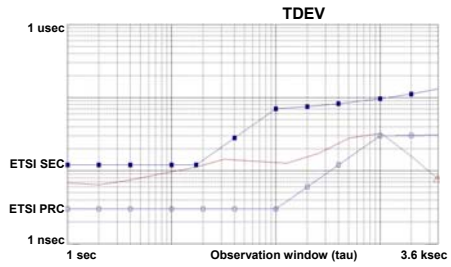
- ITU-T Rec. G.822 defines frequency accuracy in terms of slip rate objective. Assuming entire budget is allocated to one slip buffer:
  - ≤ 5 slips in 24 hrs (>98.9% of time) ⇒ Δf < 7.3 ppb* (>98.9% of time)
  - ≥ 30 slips in 1 hr (<0.1% of time) ⇒ Δf > 1 ppm (<0.1% of time)
- ITU-T Rec. G.823 defines frequency accuracy (2048 kbit/s hierarchy) in terms of MTIE:
  - Synchronous E1 : MTIE(2000s) < 2 μs ⇒ Δf < 1 ppb* (2000s)
  - Traffic E1 : MTIE(1000s) < 18 μs ⇒ Δf < 18 ppb* (1000s)
- ITU-T Rec. G.824 defines frequency accuracy (1544 kbit/s hierarchy)in terms of MTIE:
  - Synchronous DS1 : MTIE(τ > 280s) < (997+0.01τ) ns ⇒ Δf < 10^{-11} (long term)
  - Traffic DS1 : MTIE(24 hr) < 18 μs ⇒ Δf < 0.2 ppb* (86,4000s)

* 1 ppb = 0.001 ppm

Telecom Clock Characteristics

- Clock performance must be within limits set by performance masks (MTIE and TDEV)
- Performance must be viewed as a function of a time interval, not just an instantaneous metric
- These charts show plots of MTIE and TDEV for a typical 1588 master-slave setup with no network load over a 3-hour observation interval





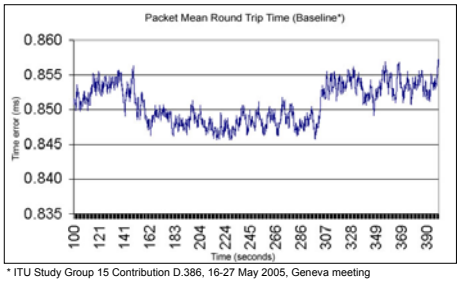
Network Simulation

- Simulation Setup
  - Native GigE line rate is assumed
  - Equal priority treatment for synchronization messages
  - Noise properties and time constants of Stratum 3E
  - 100 Hz update rate
- Cases considered
  - Case 1: 8 hops over GigE with 15% to 20% variable loading†
  - Case 2: 8 hops over GigE with 70% to 80% variable loading
  - Cases 3&4: 3 and 5 hops over GigE with 70% to 80% variable loading
  - Case 5: ADSL2 speed last mile link supporting Ethernet, with 50% to 98% variable loading
- Measurements include
  - MTIE

† Loading is not assumed to be stationary (the mean is not constant over time). This simulation supports the concept of a "busy hour"

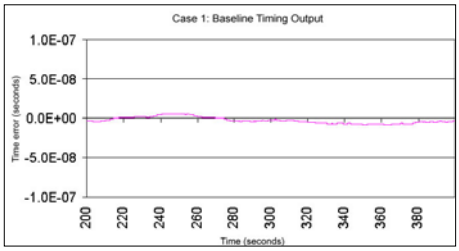
Case 1: Low Loading Packet Delay

- Loading between 15% - 20%
- Delay variation is less than 10 µs



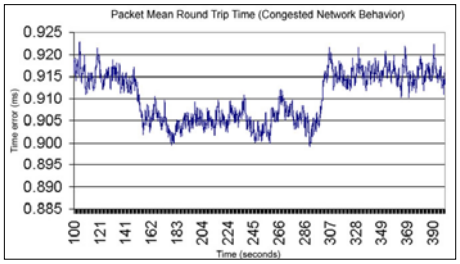
* ITU Study Group 15 Contribution D.386, 16-27 May 2005, Geneva meeting

- Output performance of derived clock is very good
- 100-second MTIE value is in the 10 ns range

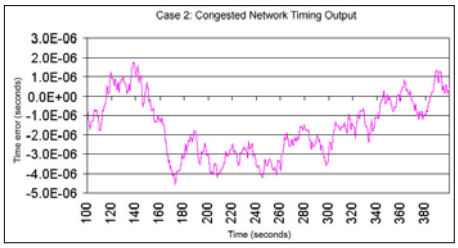


Case 2: Packet Delay in a Congested Network

- Loading between 70% - 80%
- Both mean delay bias and variability have increased
- Delay variation is less is still small (~20 µs)

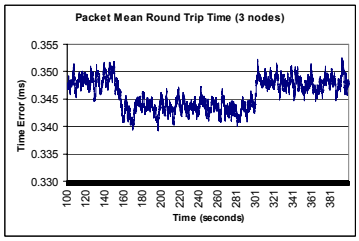


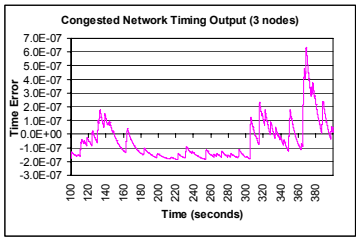
- Lower output performance of derived clock
- 100-second MTIE value is 6000 ns



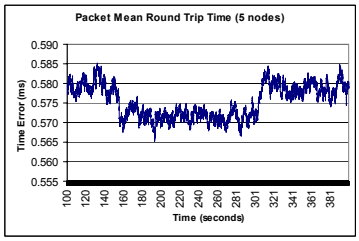
Case 3 & 4: Packet Delay with Lower Node Count

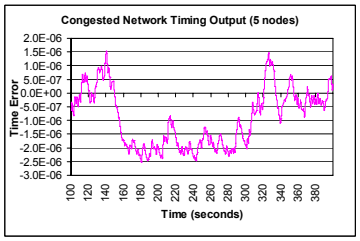
- 3 nodes, 70% - 80% loading
- Delay variation is ~15 µs
- Output MTIE is in the 800 ns range





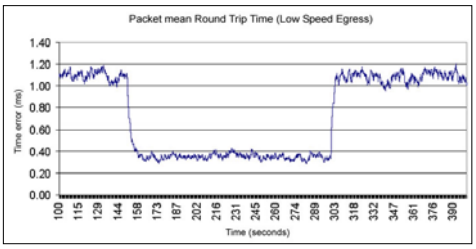
- 5 nodes, 70% - 80% loading
- Delay variation is ~20 µs
- Output MTIE is in the 4000 ns range



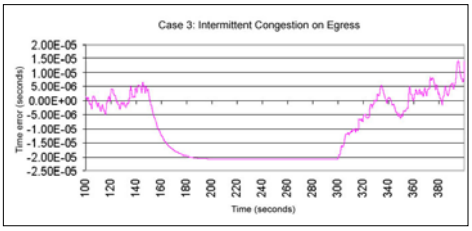


Case 5: Lower Speed Egress Packet Delay

- Simulated last mile, where link speed is typically much lower
- ADSL2 link speed is assumed
- Load variation is between 50% - 98%
- Mean RTD peaking over 1 ms



- Performance shows significant degradation
- 100-second MTIE is over 39000 ns



Summary of Simulation Results (Interim)

Number of Nodes	Load variation	Delay Variation	100-second MTIE	G.823/G.824 requirements
8 hops over Gig E	15-20%	10 µs	10 ns	Meets both G.823b/G.823c & G.824b/G.824c requirements
3 hops over Gig E	70-80%	15 µs	800 ns	
5 hops over Gig E	70-80%	20 µs	4000 ns	Meets only G.823c/G.824c requirements
8 hops over Gig E	70-80%	20 µs	6000 ns	
ADSL Last mile	50-98%	1000 µs	39000 ns	Does not meet G.823b/G.823c or G.824b/G.824c requirements

- Network congestion that varies with time significantly impairs performance
- In pure packet networks, G.823 & G.824 requirements are met in two scenarios
  - Networks with very low capacity utilization
  - <3 hops between 1588 clock and client (GigE case)
- In telco access networks where ADSL links are typically deployed, IEEE 1588 does not meet G.823 or G.824 requirements

Implementation Considerations

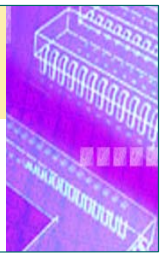
- Network congestion that varies with time significantly impairs performance
- Constraining number of hops between 1588 master and client can avoid congestion induced performance impairment
- Lower speed ingress/egress links can significantly impact time transfer
- ADSL based last mile network will certainly require reduction of hops between the master and the slave
- Other transport mechanisms (non-native Ethernet) such as xDSL(VDSL, ADSL2), WiMAX, PON, POS, DOCSIS must be carefully studied

Authors Contact Information

George Zampetti gzampetti@symmetricom.com
Vandana Upadhyay vupadhyay@symmetricom.com
Srini Bangalore sbangalore@symmetricom.com

IEEE 1588 in Telecommunications Applications

Dave Tonks
Semtech, Advanced Communications Division
Southampton,
UK



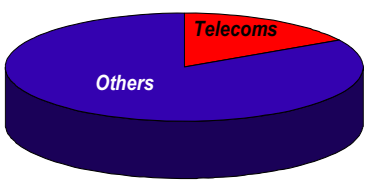
SEMTECH

Semtech Confidential – unauthorized copying prohibited

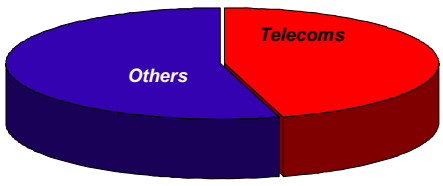
Growth of Telecom Companies in IEEE 1588 activity

This year, 21 out of 46 Companies taking part in 1588 activities are Telecom Companies

Compare this with only 6 Telecom Companies out of 36 total last year.



2004



2005

Why?

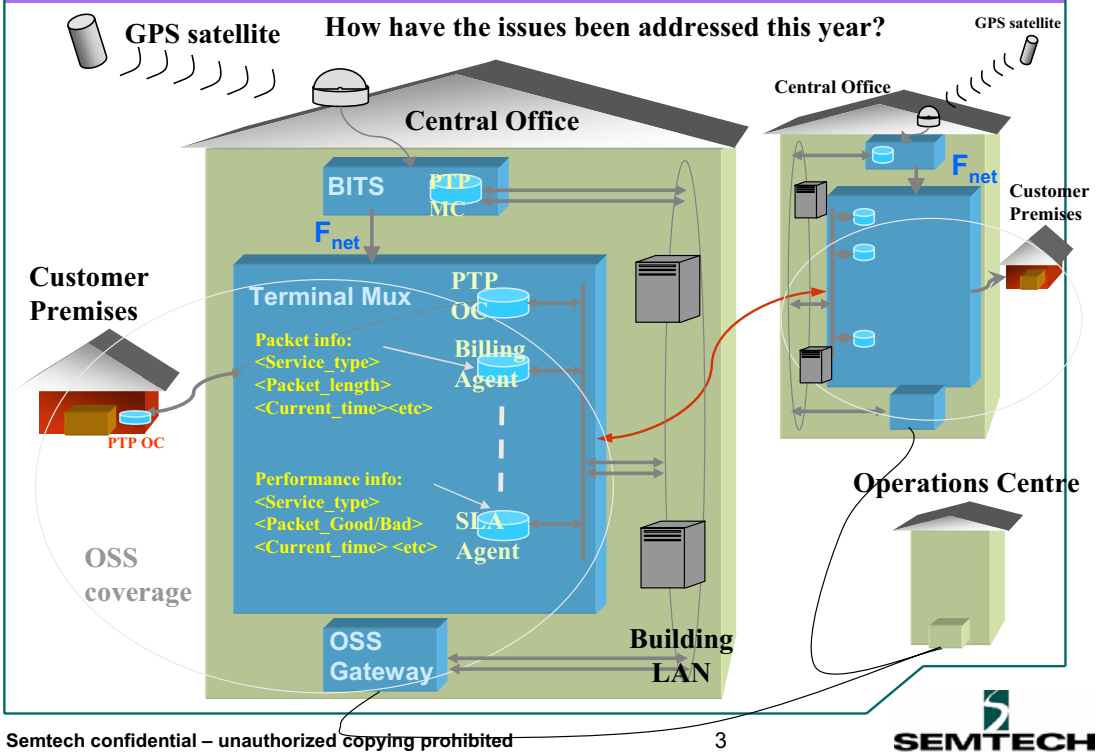
Because 1588 offers a way to solve many problems with the next generation of telecom networks, as highlighted last year.....

Semtech confidential – unauthorized copying prohibited

SEMTECH

IEEE 1588 in Telecoms:

Accurate timing to help Billing and SLA
Accurate, stable frequency for circuit emulation



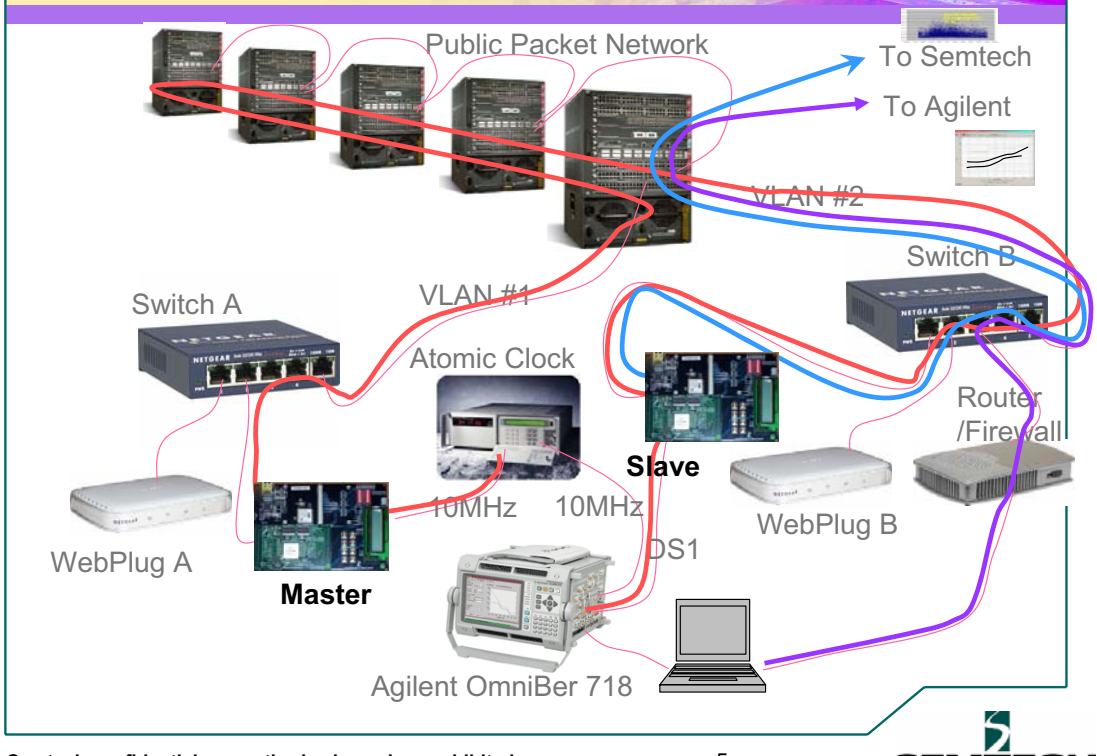
SEMTECH

Real-World Experience

- Since April 2005, Semtech and Agilent have been running a field trial on a live Metro Ethernet Network
- Semtech provided two Evaluation boards to act as IEEE 1588 Master and Slave
- Agilent provided the reference and measurement equipment
- The Master sent 1 Sync message every 2 seconds
- The Slave sent 24 Delay-Request messages every second
- TIE data was gathered for every 24 hour period and analysed to get MTIE, TDEV and Frequency Offset.

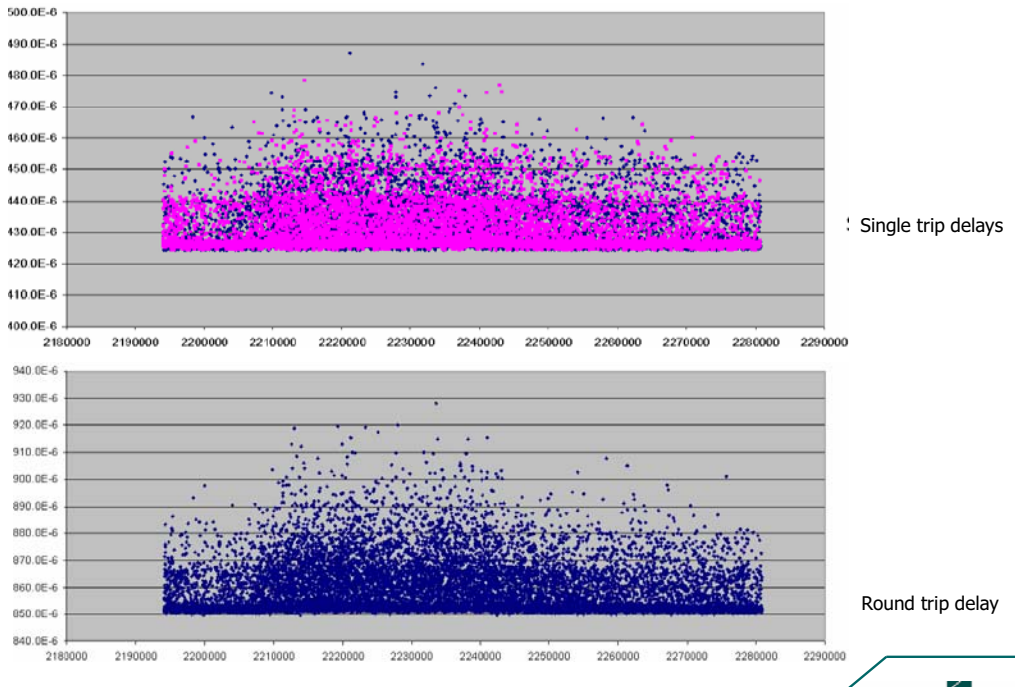
SEMTECH

Arrangement of Field Trial



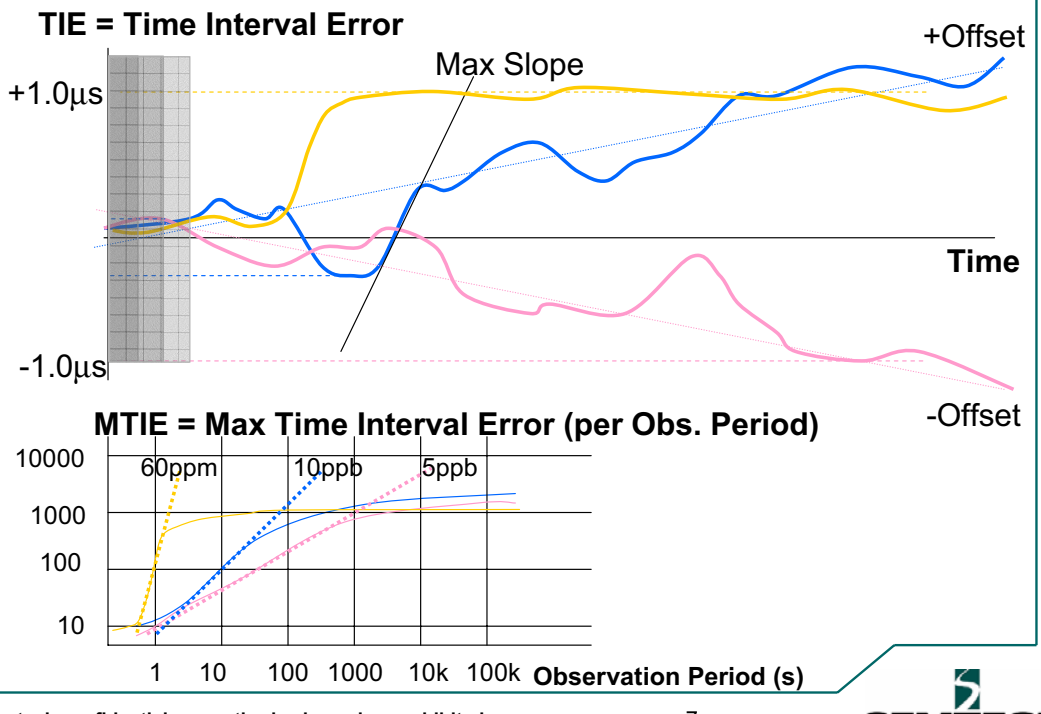
SEMTECH

Packet delays seen on the Metro Ethernet Network



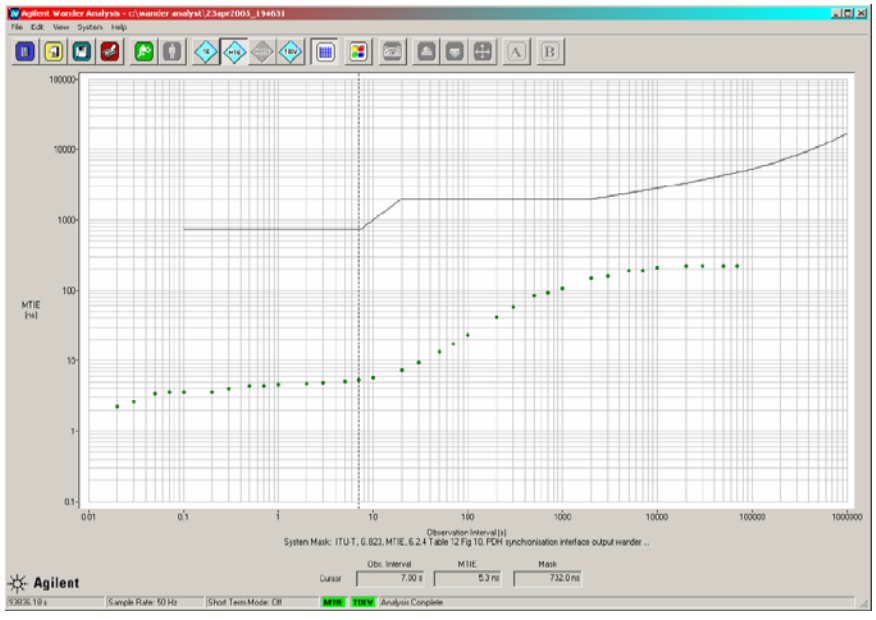
SEMTECH

For non-Telecom engineers…..what do we mean by MTIE?…



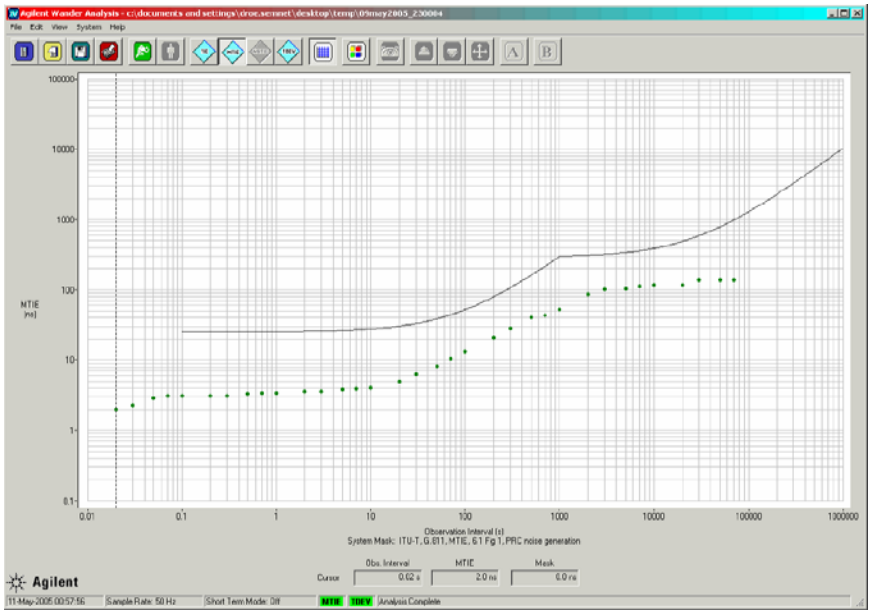
SEMTECH

OCXO Slave MTIE compared to ITU G.823 PDH Sync allowance

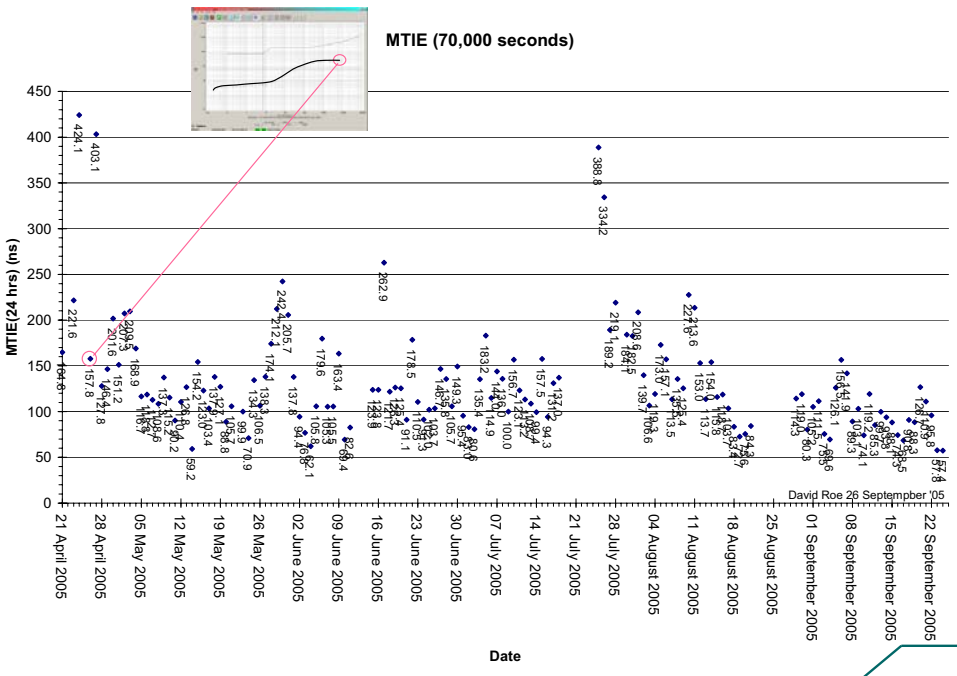


SEMTECH

OCXO Slave MTIE compared to ITU G.811 PRC allowance



OCXO Slave Daily Peak MTIE values during trial

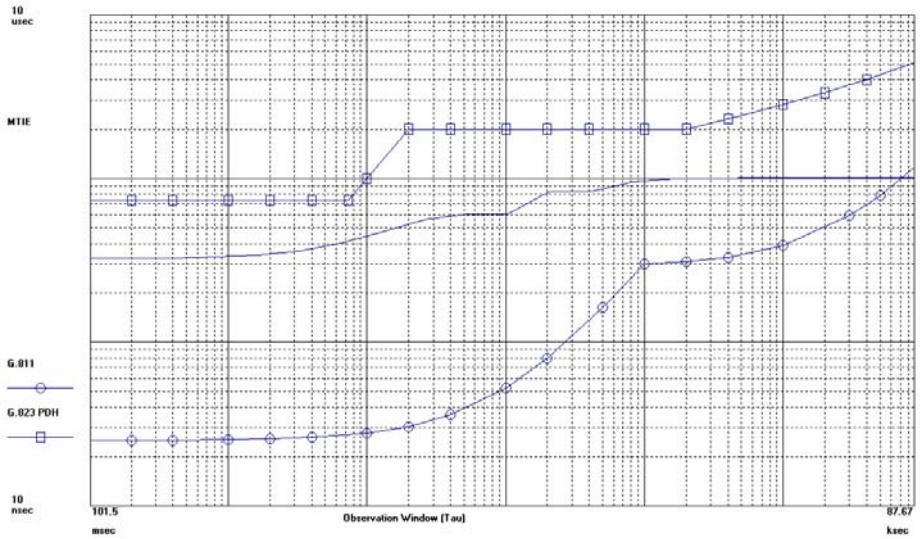


Extension to the field trial

- Last week, the field trial was extended:
  - Epoch testing proved the Master and Slave held to within 100ns.
  - A second Slave used a TCXO, in place of the OCXO used on the original Slave, to compare the price/performance ratio of the different oscillator technologies.

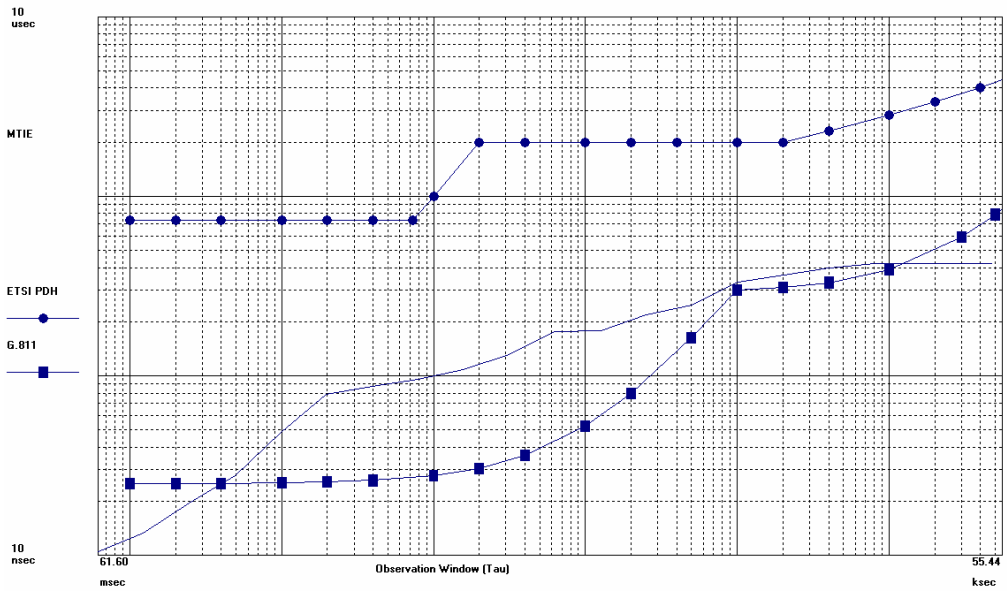
SEMTECH

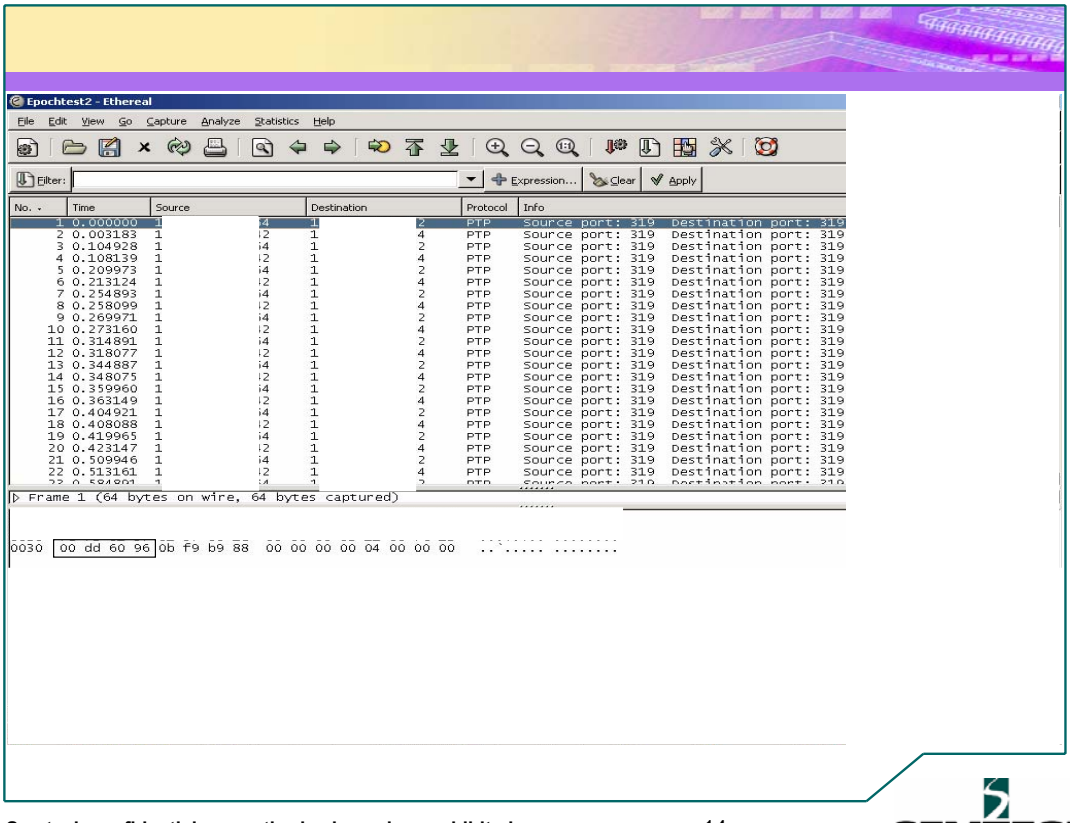
TCXO (non optimised) Slave MTIE compared to ITU G.823 PDH Sync allowance

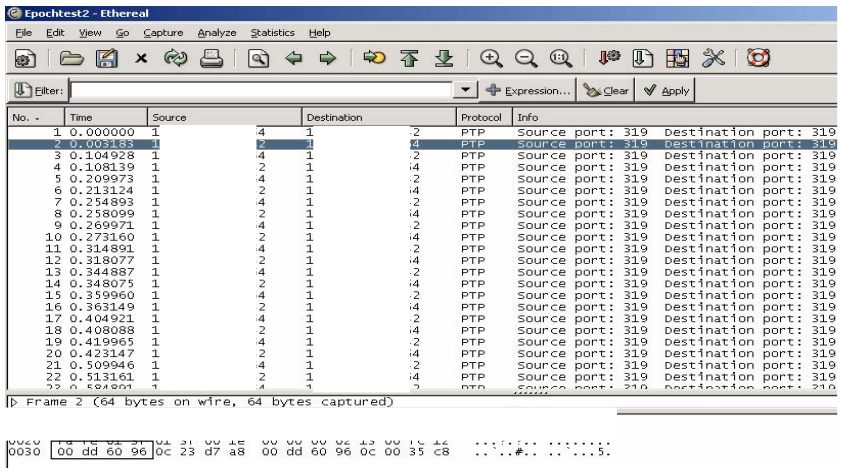


SEMTECH

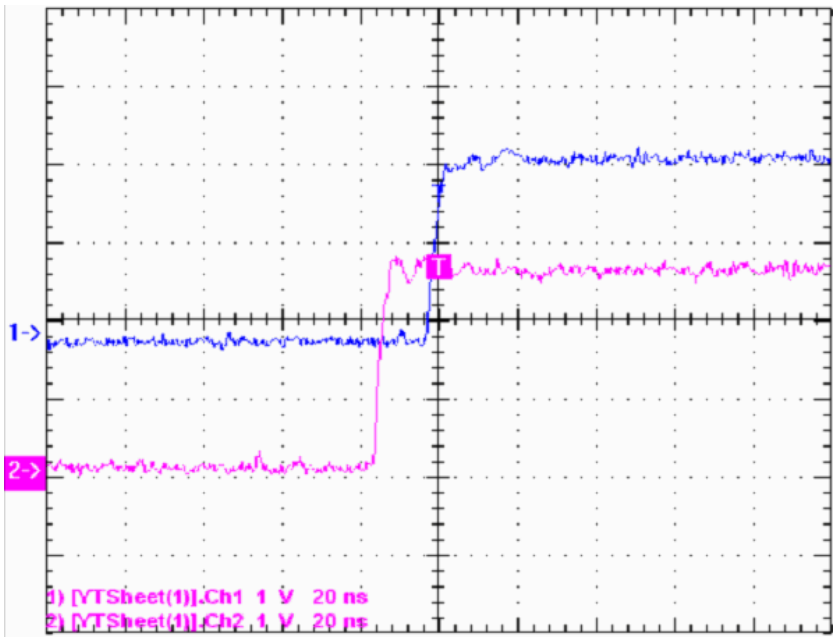
TCXO Slave MTIE compared to ITU G.823 PDH Sync allowance



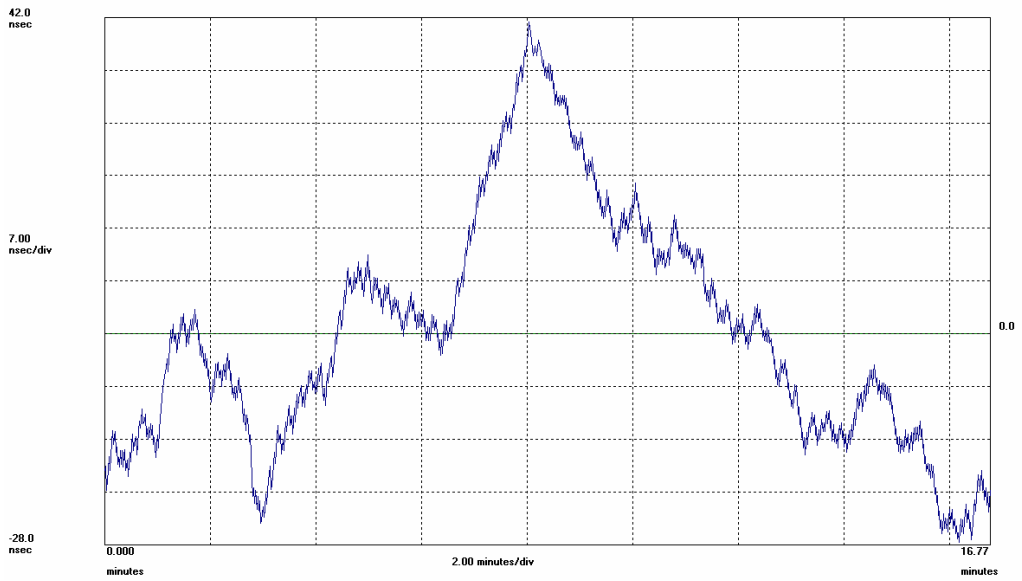




Slave 1 PPS TIE measured relative to Master

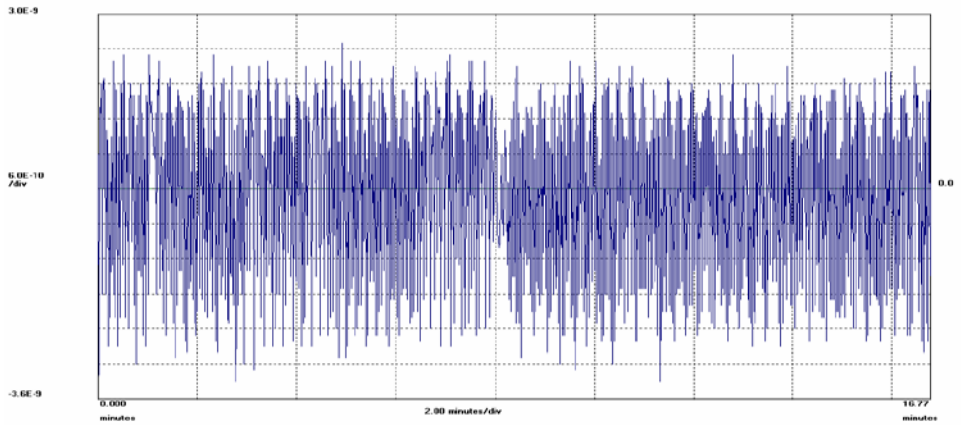


Slave 1 PPS TIE measured relative to Master



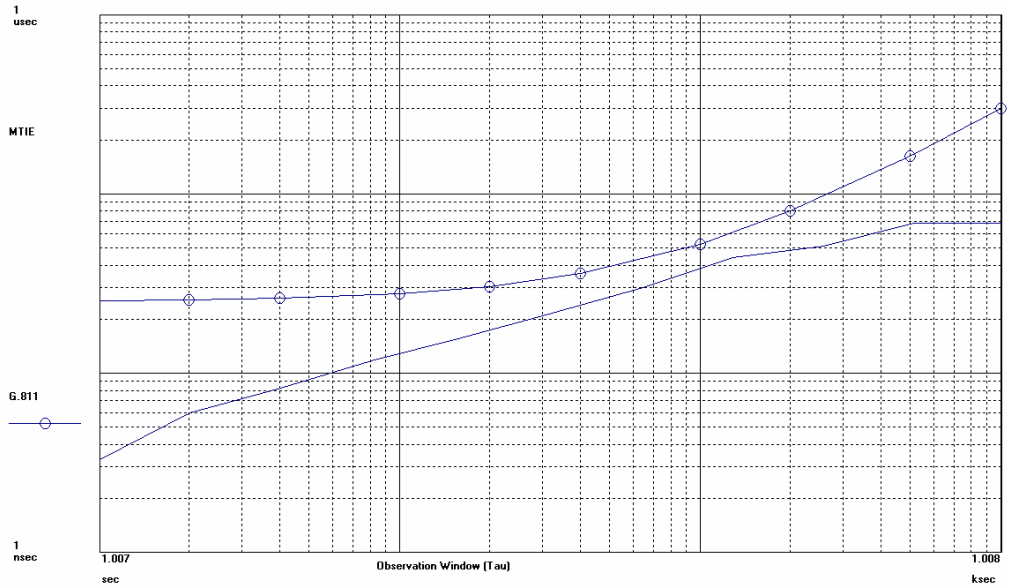
SEMTECH

Fractional Frequency Offset



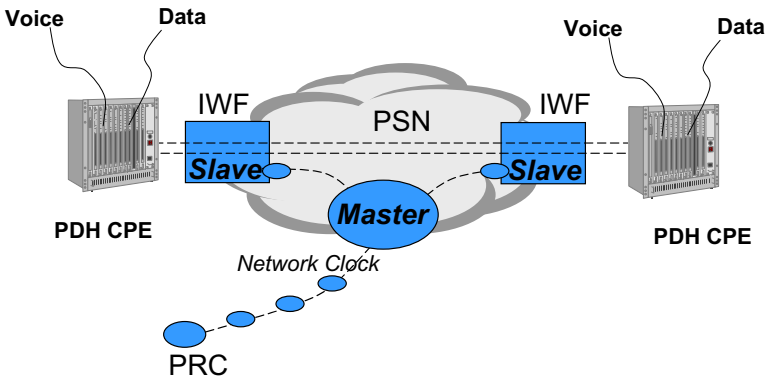
SEMTECH

Slave 1 PPS MTIE compared to ITU G.811 PRC allowance



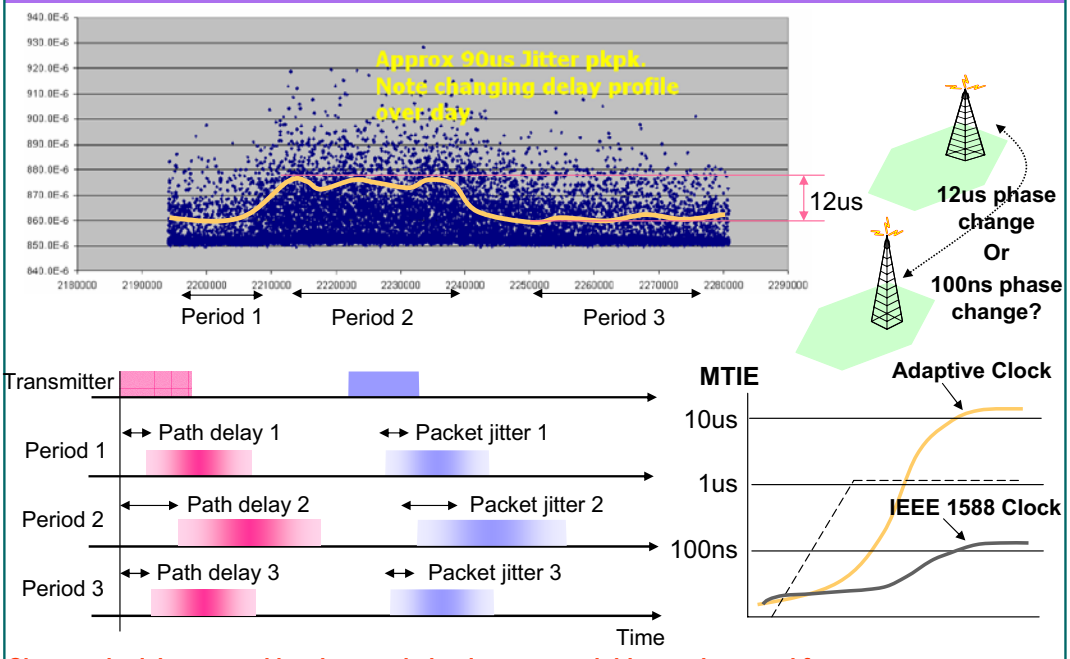
What does this mean for Possible Applications?

One example: IEEE 1588 can support CES better than can Adaptive-Clocking.....



- IEEE 1588 provides a replica of the network clock in the IWFs and allows them to operate in a network-clocked mode for best timing quality
- Studies have shown that the PDV of the traffic can be much greater if network-clocked mode is used. Same thing for packet loss.
- The IWF can now be co-located with the CPE, allowing PSN to extend to CPE.

How does adaptive-clocking cope under the same stress?



Changes in delay caused by changes in load cause much bigger phase and frequency transients in adaptive clocks

SEMTECH

IEEE 1588 supporting OSS

- Making accurate timing available at each NE allows individual packets to be monitored on ingress to, and egress from, the network.
- This allows accurate latency measurement and helps maintain SLA-compliance.
- Also allows billing by service, rather than the limited, inflexible, bundling of today. This allows more individual choice for the customer and raises more revenue for the network provider.

SEMTECH

Summary

- IEEE 1588 has increased its profile in Telecoms over the last year. Nearly half of the regular contributors are from Telecom companies. This is because IEEE 1588 promises to solve the timing problems of the forthcoming packet networks.
- Experience with a long-term field trial shows that excellent accuracy and stability is actually available, holding true to the promise. This trial used unicast messages, sent at a faster rate, showing that the new PAR work is very relevant.
- The other issues which were highlighted last year are being worked out and will allow IEEE 1588 to be used in many telecom applications.
- Although Boundary Clocks or maybe Transparent Clocks may eventually become available in a future telecom network, it is unlikely that early designs of Ordinary Clocks will find applications across the different industries, and if a common standard is not made available, this will be impossible.
- Finding a good balance between the sometimes conflicting requirements of the different industries will be imperative if such a common standard is to be arrived at.

Semtech confidential – unauthorized copying prohibited

SEMTECH

Synchronizing IEEE 1588 clocks under the presence of significant stochastic network delays

Carmelo Iantosca[(1)], Christoph Heitz[(1)], Hans Weibel[(2)]

[(1)]Institute for Data Analysis and Process Design, Zurich University of Applied Sciences, CH-8401 Winterthur

[(2)]Institute of Embedded Systems, Zurich University of Applied Sciences, CH-8401 Winterthur

The clock synchronization method described in IEEE 1588 is often used in closed and application specific networks, e.g. for distributed measurement or automation and control applications. These networks can be equipped with special switches working as boundary or transparent clocks. In non IEEE 1588 aware switched networks such as Intranets, Metro Ethernets, MPLS or EFM Networks, the method has to cope with uncorrected stochastic message transit delays.

We investigate the problem of synchronizing clocks in the presence of stochastic delays of PTP messages caused by queuing in switches. Especially in the case of heavy network traffic, queues may be filled and can lead to substantial delay of PTP messages. The aim of the presented approach is to ensure synchronization even with such large perturbations.

Measurements in a simple laboratory switched fast Ethernet network showed that transit delays in the order of magnitude of some milliseconds may arise under heavy traffic. A simple synchronization algorithm fails under these circumstances.

A synchronization algorithm was developed that is based on a generic statistical approach and explicitly takes into account the stochastic properties of the transit delay of PTP messages both in the case of delay and in the case of unhindered message flow. By using robust statistical estimation methods it is possible to estimate the drift and the time offset between master and slave reliably, even under strong stochastic perturbations.

With this approach, time synchronization in already existing networks can be improved significantly.

1. Introduction

According to IEEE 1588, a slave clock calculates the one way delay in order to take the Sync message transit delay into account. In Ethernets, the delay is not constant. Its variation is caused by different mechanisms. Transceivers and hubs cause a relatively small variance while store and forward devices like switches may delay messages significantly depending on their load. One approach to handle this problem is to use special switches like Boundary Clocks or Transparent Clocks. Switches in telecom or corporate networks however do not support IEEE 1588. An obvious approach is to collect measurement data over a longer period of time and try to extract offset and phase change as good as possible by means of statistical methods.

2. Synchronizing under stochastic delays

2.1. General

We consider a slave clock getting messages from a master clock. Synchronization is made possible by comparing the time stamp of the master clock at send time with the time stamp of the slave clock at receive time and vice versa. If the delays for the messages would be deterministic, a single message exchange master – slave – master would be sufficient for synchronizing. Usually, however, the transit delay for a packet is subject to stochastic variability due to different reasons. This makes the interpretation of the master's time stamp much more difficult. Synchronization has to take this uncertainty into account.

There are several different sources of variability. All these sources are stochastic in the sense that they cannot be precisely predicted. On the other hand, their stochastic properties are rather different:

1. Network configuration: The network may change due to physical changes or to re-configuration after breakdown of a component. This may lead to sudden changes of the transit delay.
2. Jittering: The transit delay for a packet is subject to small variations in packet processing time within hardware components. The order of magnitude of the processing time variation is in the range of one microsecond for a switch as long as no queuing occurs, and some tens of nanoseconds for hubs.
3. Queuing effects in switches: The synchronization message may be delayed in a queue, leading to random queuing delays which are in the order of magnitude of some tens up to some hundreds of μs.

In the following, we study the stochastic nature of the delays in depth. This analysis holds for all messages within a network, in particular for Sync messages and Delay_Req messages.

Each source of variability leads to uncertainty of the received information. On the other hand, the stochastic properties of this uncertainties are known. The general idea of this paper is to use this information in a consistent way for getting a reliable estimation of the master time and thus can be used for synchronization.

2.2. Statistical properties of stochastic delays

The physical delay D that a message encounters consists of different contributions:

$$D = D_{net} + D_q + \varepsilon \qquad (1)$$

where D_{net} stands for the delay time without taking into account jittering nor considering queuing. Thus, D_{net} is equivalent to the average transit time when no queuing occurs. The value of this variable is determined by the network configuration (e.g. cable lengths, processing properties of switches, …). The term D_q describes a possible delay due to queuing, and ε is a jittering component.

Each packet encounters a specific delay. We denote the delay of packet i (i=1,2,…) by D(i).

In the subsequent paragraphs, the different contributions are described in a statistical way.

2.2.1. Network configuration

The network configuration determines the value of D_{net}. It is not known but is constant over time, except when the network configuration changes. Thus, we can describe D_{net} as a random variable with the following properties:

- $D_{net}(i) = D_{net}(i-1)$ with probability p, where $p \approx 1$.
- $D_{net}(i)$ gets a new value with probability 1-p

The probability p is very small. If we consider a Sync message being sent regularly all two seconds, and a rate of network configuration changes of one per day in the average, then there is one change per $24 \cdot 60 \cdot 30 = 43200$ messages, which results in a probability of $p = 1-(43200)^{-1} = 1- 2.3 \cdot 10^{-5}$.

2.2.2. Jittering

Every packet is subject to jitter due to not exactly constant packet processing time. Note that the *average* processing time is already subsumed by D_{net}; the jitter variable ε does only account for the deviation of the processing time from the mean processing time. Thus, we can assume that $E(\varepsilon) = 0$, where E() denotes the expectation value of a random variable.

Furthermore, we assume that ε has a variance σ^2. Subsequent realizations are iid (independent identically distributed).

For dealing with the effect of jittering, it turns out that only the variance of the jitter effect has to be known. This value is more or less constant for a given switch. It is known a priori or can be measured.

2.2.3. Queuing delays

The queuing contribution D_q accounts for a possible delay if a packet encounters a queuing device that is already processing another packet such that the packet is placed in a buffer prior to processing. Consistent with the usual terminology in queuing theory, we call the processing unit "server", and the buffer "queue".

The queuing effect is zero if the packet finds the server idle (note that the processing time of the server is accounted for by D_{net} and ε). Let ρ be the probability that the server is busy (i.e., ρ is the utilization of the server, $0 \le \rho \le 1$).

If the Sync message finds the server busy, then the message has to wait until the prior packets are processed. This waiting time is variable. If the Sync packet has highest priority and can overtake lower priority packets, the waiting time corresponds to the remaining processing time of the currently processed packet plus the send time for this packet and, possibly, other packets that are processed but still not sent. If the Sync packet has not highest priority, additionally the processing time of already queued packets has to be taken into account. In each case, the waiting time can be described by a random variable W whose distribution is unknown in most cases.

Maximum length packets in Ethernets have a minimal processing time of 122 µs, while minimum packets have a processing time of about 5 µs. Thus, typical average values for W are in the order of magnitude of 1…1000 µs.

The queuing delay D_q can be described by

$$D_q = B \cdot W$$

where B is a binary random variable (B=1 with probability ρ, and B=0 with probability $1-\rho$).

The queuing delays for subsequent Sync messages are not independent since the network load usually has strong temporal correlation: Heavy traffic situations tend to stay for longer time (e.g. download time of large files, or using the network for streaming services). However, this correlation only affects the variable B: If Sync packet i finds the server busy, then the probability of packet i+1 arriving at a busy server is increased. In statistical terms: B(i) and B(i+1) are (positively) correlated.

On the other hand, since the arrival of the Sync packet is uncorrelated to the arrival of the packets that are currently under work at the server, subsequent W's are uncorrelated: the series W(i) (i=1,2,…) is iid.

2.3. Improved synchronization algorithm for stochastic delays

A time synchronization working with stochastically delayed Sync messages has to take into account the stochastic properties of the delays. The better the algorithm takes the (stochastic) properties of the delay into account, the better it will work.

For synchronizing the slave clock, a good estimate of the clock difference between master and slave must be provided. This estimate is updated each time a new Sync packet is received.

In most cases, the correction of the slave clock is performed immediately according to the estimate. In order to keep the following discussion simple, we assume that the slave clock is *not* corrected upon the receipt of a

Sync message. This simplifies the equations. However, the same approach can be used for updating after each Sync message.

2.3.1. Master and slave time stamps

In general, the master clock and the slave clock run at different rates which leads to a drift between master and slave:

$$t^{(slave)} = t^{(master)} + \alpha \cdot t^{(master)} + \beta \,. \tag{2}$$

with an unknown drift term α, and β being the difference of the two clocks for t=0. In the following, we assume that α is constant. Later we will relax this condition.

The offset between the master clock and the slave clock is given by

$$offset = \alpha \cdot t^{(master)} + \beta \,. \tag{3}$$

When a Sync message and its Follow-up message have been received, $t_{sent}^{(master)}$ and $t_{received}^{(slave)}$ are given. The receiving time of the Sync message is given by

$$t_{received}^{(slave)} = t_{sent}^{(slave)} + D \tag{4}$$

where $t_{sent}^{(slave)}$ is the slave clock time when the master sends the Sync message, D is the transit delay time, and $t_{received}^{(slave)}$ is the slave clock time when receiving the Sync message.

If α and β are known, with Eq. (2), a time stamp from the master clock can be transformed into a time in slave's time reference:

$$t_{sent}^{(slave)} = t_{sent}^{(master)} + \alpha \cdot t_{sent}^{(master)} + \beta \,. \tag{5}$$

Combining Eqs. (4) and (5), one can write

$$t_{received}^{(slave)} = t_{sent}^{(master)} + \alpha \cdot t_{sent}^{(master)} + \beta + D \tag{6}$$

Combining Eqs. (6) and (1), one gets

$$t_{received}^{(slave)} = t_{sent}^{(master)} + \alpha \cdot t_{sent}^{(master)} + \beta + D_{net} + D_q + \varepsilon \tag{7}$$

For Sync message i, this reads

$$t_{received}^{(slave)}(i) = t_{sent}^{(master)}(i) + \alpha \cdot t_{sent}^{(master)}(i) + \beta + D_{net}(i) + D_q(i) + \varepsilon(i) \tag{8}$$

For a one-way connection master→slave it is not possible to distinguish between β and D_{net}. Consequently, from the received information only the sum

$$S = \beta + D_{net}$$

can be determined. For determining β, one needs additional messages from slave to master (Delay_Req messages) which allows then to separate the two contributions.

For the new synchronization method, we construct two specific estimators: First, an estimator $\hat{\alpha}$ for the drift parameter α is introduced. Second, an estimator $\hat{S}$ for S is constructed.

2.3.2. Estimator for drift

In this section it is shown how the drift estimator is constructed. We assume that we get a series of Sync messages with time stamp pairs $t_{sent}^{(master)}(i)$ and $t_{receive}^{(slave)}(i)$, where i=1,2,3,…

If we regard the difference Δ(i) between these two time stamps

$$\Delta(i) = t_{sent}^{(master)}(i) - t_{received}^{(slave)}(i) \,,$$

we get

$$\Delta(i) = \alpha \cdot t_{sent}^{(master)}(i) + \beta + D_{net} + D_q + \varepsilon$$

Note that D_{net} can be assumed as constant for most time periods, ε is a small iid random variable, and D_q is a random variable that either is zero or has a large positive value. Since the Sync messages are sent in equal time intervals, one can set

$$t_{sent}^{(master)}(i) \approx i \cdot Dt$$

where Dt is the time interval between subsequent Sync messages.

Plotting Δ(i) against i leads to a pattern like shown in **Fig 1**. The drift of the slave clock with respect to the master clock can be seen clearly, as well as the large positive peaks when queuing occurs.

From this time series, the drift is estimated by the following simple procedure: A set of M·n consecutive measured differences Δ(i), i=1,2,…, M·n is partitioned into M frames. For each frame j=1,…,M, the *minimum* value of the Δ(i) in this frame is denoted by y_j, its time stamp $t_{sent}^{(master)}(i)$ is denoted by x_j. From the M (x_j,y_j) such pairs, the drift is estimated by linear regression. Note that for this estimation, the value of S has not to be known.

Taking the minimum value of Δ(i) in each frame makes sure that large delays due to queuing are not considered (as long as there is at least one packet in the frame that did not encounter queuing).

This drift estimation can be updated for each arriving packet. Alternatively, it can be used for the next M·n packets. Then, a new drift estimation is performed.

2.3.3. Estimation of S

For the estimation of S, we consider the series Δ(i), corrected by the drift term. This leads to

$$\Delta_{corr}(i) = \Delta(i) - \hat{\alpha} \cdot t_{sent}^{(master)}(i) \approx \beta + D_{net} + D_q + \varepsilon$$

The left side is a measured quantity, the right hand side is composed by the constant value

$$S = \beta + T_{net}$$

and the two random variables D_q and ε.

In the following, we derive an estimate of S by using the corrected sequence Δ_{corr}(i). Combining this estimate with the analogous estimate for the backwards direction, a synchronization is possible.

When estimating S under stochastic variations requires some sort of averaging. A widely used method is the exponential smoothing where an estimate $\hat{S}$ of S is constructed by setting

$$\hat{S}(i) = \hat{S}(i-1) + (1-\gamma) \cdot \left(\Delta_{corr}(i) - \hat{S}(i-1)\right).$$

Here, $\hat{S}(i)$ is the estimate that is calculated when packet i arrives, where $\hat{S}(i-1)$ is the estimate that has

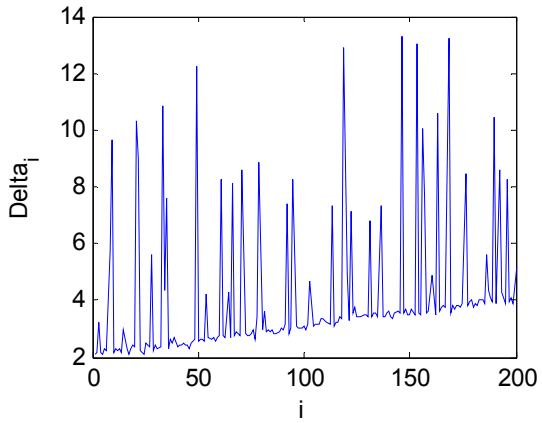


Fig 1: Schematic plot of i against Δ(i) for a server utilization of about 80%.

been calculated one step before. The second term at the right hand side measures the difference between the current measured $\Delta_{corr}(i)$ and the old estimate $\hat{S}(i-1)$. This is used as a correction term for $\hat{S}(i-1)$. The constant γ is a factor that denotes how much effect the current value $\Delta_{corr}(i)$ has on the estimate $\hat{S}(i)$. For γ=0, the old estimate is not used, and we end up with

$$\hat{S}(i) = \Delta_{corr}(i).$$

This means, no smoothing takes place. On the other hand, when setting γ=1, then

$$\hat{S}(i) = \hat{S}(i-1).$$

That means, the current measured value does influence at all the estimate. In practice, a value of $\gamma \approx 0.9$ is often used. In this case, a history of about the last 10 samples is taken into account[1].

This estimator is well suited for the case of symmetric deviations which are not too large. However, for the problem of estimation of S under strong stochastic queuing effects, this estimator is not suitable since the occurrence of a queueing delay leads to a large difference $\Delta_{corr}(i) - \hat{S}(i-1)$ and thus, to a significant change of $\hat{S}(i)$.

The problem of outliers having a large effect on statistical estimates is well known in statistics and has led to the so-called "robust statistics" [3]. The principal idea there is to construct estimators that are not sensible to outliers.

For the present task of estimating S, the outliers can only occur on the positive side since queuing only can lead to longer delays, never to shorter delays. Thus, when constructing a robust estimator for S, only large positive values of the difference $\Delta_{corr}(i) - \hat{S}(i-1)$ should be filtered out. On the other hand, negative values should not be damped.

An estimator that makes this job is given by

$$\hat{S}(i) = \hat{S}(i-1) + (1-\gamma) \cdot g\left(\Delta_{corr}(i) - \hat{S}(i-1)\right)$$

where g() is a function defined by (see **Fig 2**)

$$g(x) = \begin{cases} x, & x < 3\sigma \\ 3\sigma, & x \geq 3\sigma \end{cases}.$$

This means, g(x) is identical to x as long as x is negative or smaller than a limiting value 3σ. When x is larger than 3σ, then g(x) remains at the maximum value of 3σ.

For σ, the standard deviation of the jitter is used. For small differences in the order of magnitude of the usual jitter, the robust estimator works exactly like the classical exponential smoother. However, when the observed time difference Δ(i) deviates from the current estimate $\hat{S}(i-1)$ more than three times the standard deviation of the jitter, this value enters the formula for the new estimate $\hat{S}(i)$ with a smaller weight. Such large positive outliers are strongly damped, and the estimator gets robust against such outliers.

[1] Note that, strictly speaking, all past values do influence the current estimate. However, the more a sample lies in the past, the less its influence is. The value of 10 samples is an indication of the length of the memory of this estimator.

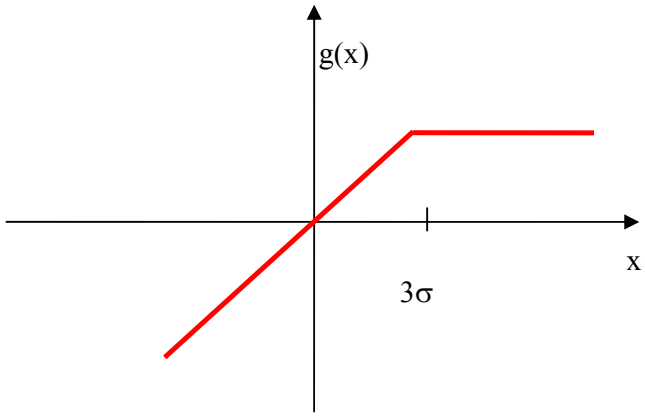


Fig 2: Weighting function g(x) for the robust exponential smoothing.

2.3.4. Leap detection

Changes of network configuration may lead to sudden changes of the transit delay. A synchronization algorithm has to be able to deal with such a situation. In such a case, the value of T_{net} changes from one Sync message to the next, leading in a change of $\Delta_{corr}(i)$ that is large compared with the jitter. However, large changes of $\Delta_{corr}(i)$ can also be found if queuing occurs. The algorithm must be able to distinguish between these two cases. It should update the offset S if there is a true network change, but it should not do so if there is only queuing.

For the leap detection, a detector has been constructed which is based on the statistical properties of the delay time due to queuing: Subsequent queuing delays generally have large differences because they find the server in different states. Even if two subsequent Sync messages find the server in a busy state, the queuing delay varies because the remaining processing time of the packet which currently is processed differs. If the order of magnitude of the queuing delay is much larger than of the jitter, subsequent values of $\Delta_{corr}(i)$ differ more than the jitter's standard deviation in nearly all cases.

For leap detection, the difference of two subsequent values $\Delta_{corr}(i)$ and $\Delta_{corr}(i+1)$ is regarded. If this difference is lower than 3σ, where σ is the standard deviation of the jitter, then it is assumed that T_{net} has changed, and S is updated.

If queuing delays occur only rarely ($\rho \approx 0$), then a change of D_{net} is detected at the second packet. For larger utilization, it takes longer.

The probability that two subsequent packets are *not* delayed by queuing is given by $(1-\rho)^2$. Thus, on average, it takes $1+(1-\rho)^{-2}$ packets until the change of network configuration is detected. For a utilization of $\rho=0.5$, this results in 5 packets, for utilization of 90% ($\rho=0.9$), it takes 101 packets on the average.

Thus, for the case of moderate utilization, a change of network topology is detected quickly.

2.4. Empirical investigation of statistical properties

A simple test set-up was created consisting of several PCs working as load generator for a conventional switch without boundary clock functionality. The load sink consisted of a computer connected to the other side of the switch over a hub. Additionally, a PTP master clock was sending Sync messages to a PTP slave using the same switch and hub. In **Fig 3** the setup is shown schematically.

The transit delays were measured with the HRTA (High Resolution Timing Analyzer, a measurement tool developed by the Institute of Embedded Systems) for different load conditions.

In **Fig 4**, a histogram of the measured delay times under no-load conditions are shown. These delay times correspond to the jitter of the switch. It can be seen that the transit delays can be described by a normal dis-

ter's rate. However, for this study the slave's rate was kept constant during the whole simulation time. The slave has a synchronization method that corresponds to the above described one which is applied both on the Sync messages and the Delay_Req messages. For comparison, also a simpler synchronizing method has been implemented.

The jitter was modelled by a normally distributed random variable with standard deviation σ=39 ns. The processing time of the switch was 17762 ns for Sync packages and 119818 ns for data packages (this processing times are means of the measured distributions for the two different package types in the experimental test setup).

3.2. Using simple method for synchronizing

As a reference we use a typical PTP implementation which was designed to be used in networks consisting of hubs and Boundary Clocks. This implementation encounters relatively small variations in path delay and has to be able to quickly respond to changes in frequency or topology. For this reason, the method takes only a small portion of the past (some ten measurement values) into account.

Both the drift and the offset are updated without taking into account the occurrence of queuing delays. The offset is estimated with a PI controller which is similar to exponential smoothing. The drift estimation is made by averaging over several Sync messages. Under usual conditions, after each Sync message, the slave clock is corrected. A leap detector is used which is based on comparing the time stamp of the received packet with the slave clock. If an unusually large deviation is recorded, the clock of the slave is reset. This procedure is adequate if there are no significant queuing delays.

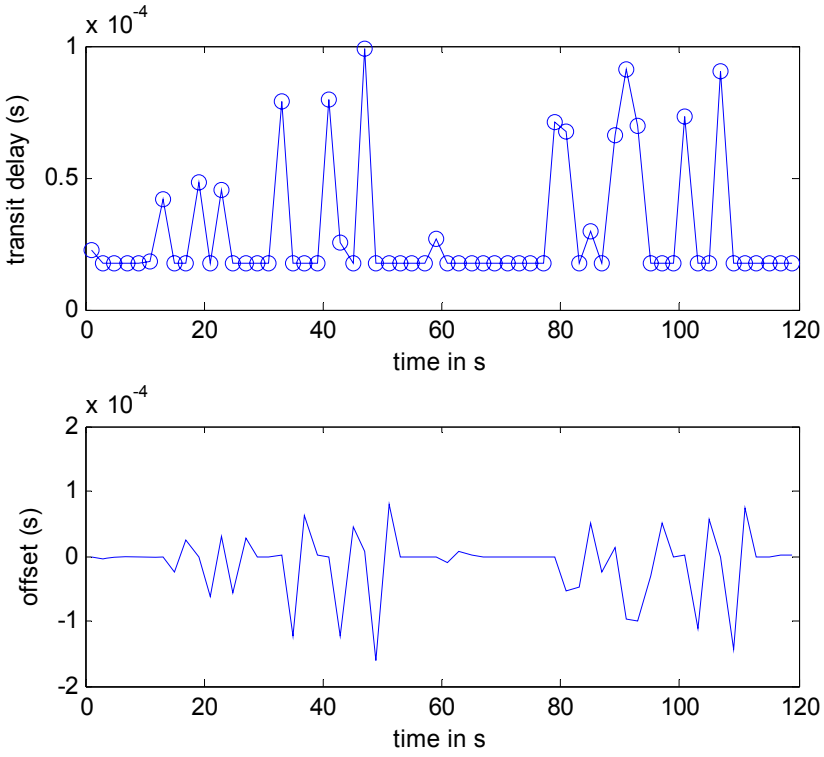


Fig 5: Simple synchronization method. Top: Delay time of Sync messages. If queuing occurs, large delay times in the order of 10...100 µs appear. The jitter variability of about 40 ns cannot be seen in this plot. Bottom: Time deviation of master clock and slave clock.

In **Fig 5**, the results of the synchronization are shown. If queuing occurs, then the delay time is much larger (order of magnitude of 10..100 µs) than the usual deviation (order of magnitude of 40 ns). Thus, the leap detection assumes a change of network topology and resets the slave clock. Additionally, the slave clock rate is changed. However, one of the next Sync messages hits an empty server and shows a low transit delay which again activates the leap detection, resulting in a second reset of the slave clock and the clock rate. Because the slave clock rate also is changed, it takes more than one Sync message before the slave clock is synchronized again.

3.3. Improved method

The improved method for synchronization leads to the results as shown in **Fig 6**. For synchronization it is assumed that the deterministic part of the offset is known.

As can be seen, the delay time estimation of the slave clock is robust against the occurrence of large queuing delays. Consequently, the slave clock is not reset. The deviation between master and slave is kept in the range of some ns which is given basically by the jitter. The deviation is less than the jitter of 40 ns because

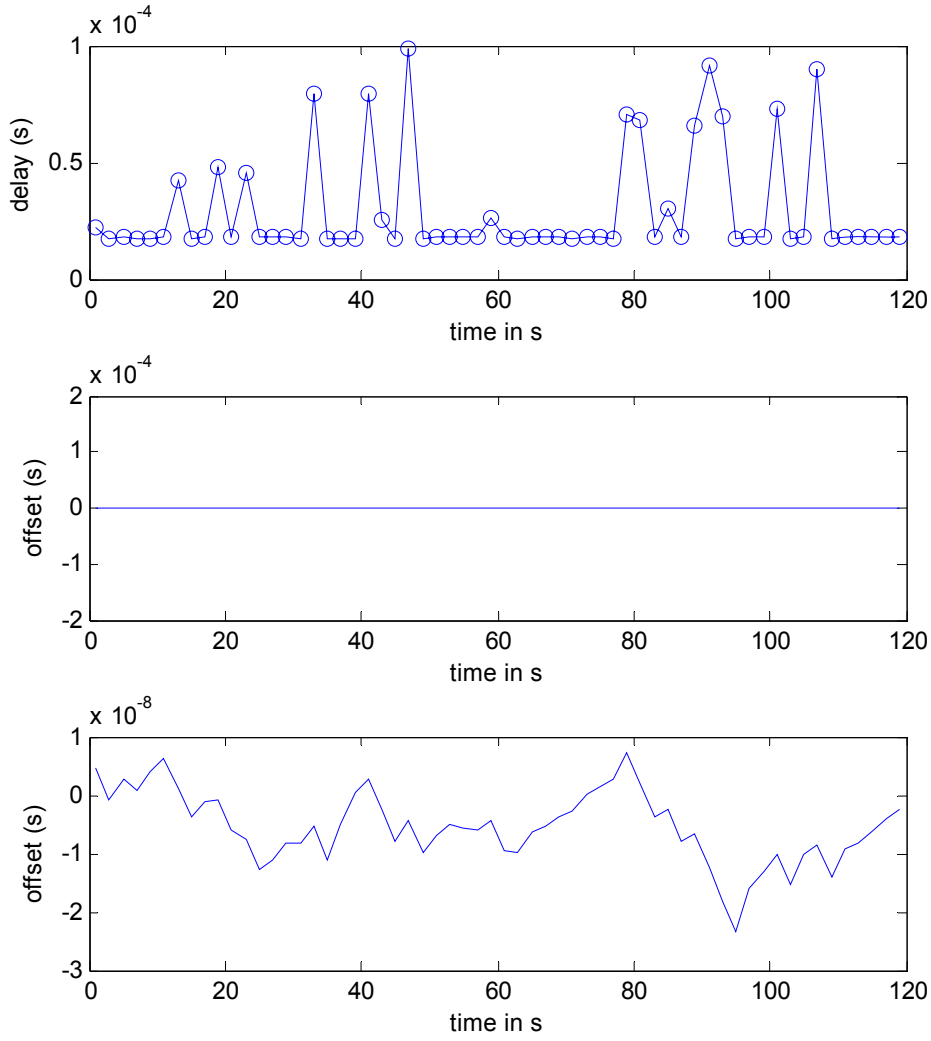


Fig 6: Improved synchronization method. Top: Delay time of Sync messages. Center: Time deviation of master clock and slave clock with the same time scale as in Fig 5. Bottom: Time deviation of master clock and slave clock, zoomed.

of the averaging which is performed in the estimators.

3.4. Discussion

As can be clearly seen in the last section, the improved method for synchronization is able to deal with the occurrence of large queuing delays, due to the robust estimation of drift and offset. However, the presented study did use some simplifications that have to be commented on:

- Only Sync messages were regarded (master → slave). From this one-way communication, only the sum of the offset and the network delay can be determined. For a synchronization, however, the value of the offset has to be known. Usually, this is made by additionally measuring the delay time for messages slave→ master (Delay_Req messages) and calculating a round trip time. By using this time, one can separate between the two components β and D_{net}. With the presented approach of robust estimation, this can be done similarly. Since the Delay_Req messages are also subject to queuing effects, they have to be processed like the Sync messages, leading to an additional estimate for the value of S for a master→ slave message. The true offset between master and slave can be calculated from these two values of S. The accuracy of the offset estimation is in the same order of magnitude as the accuracy that has been determined for the Sync messages alone. However, since Delay_Req messages are send much less frequently than Sync messages, the properties of the exponential smoother have to be adopted accordingly.
- The drift of the slave with respect to the master has been chosen as a constant. In reality, this is not the case. This will reduce the accuracy in comparison to the one observed in the present study. The presented algorithm is constructed for an iterative drift estimation and is able to follow changes in drift as long as these changes have a lower time constant than the time period needed for drift estimation.
- In order to apply the robust estimator for the offset, the standard deviation σ of the jitter has to be known. Note, however, that this value needs not to be known exactly. For proper working of the estimator it is important that the level of the robust estimator is set larger than σ such that jitter variations go into the smoothing unchanged, and small enough that typical queuing delays are heavily damped. Since the jitter is in the range of ns, and queuing delays are in the range of μs, the difference between these two values is very large such that the setting of the level of the robust estimator is not critical.
- The jitter may also be estimated from the measured time series. In this case, no apriori knowledge on the magnitude of the jitter has to be available. This additional feature will be worked out in further research.
- A change of master clock manifests itself very similar as a change of network configuration. In both cases, the value of S changes suddenly to a new value. Thus, from the side of data processing on the slave there is no difference. When evaluating the Delay_Req messages, it gets clear which effect caused the change of S.

4. Conclusions

In the present paper we describe a new method for synchronizing IEEE 1588 clocks under the presence of significant stochastic disturbances generated by queuing effects in switches or similar devices. For this, robust estimators for drift and offset are used that incorporate concepts of robust statistics. In particular, a robust exponential smoother has been constructed that is able to deal with arbitrary large delay times. The only parameter that has to be known when using this estimator is the order of magnitude of the jitter.

The concepts presented in this paper are based on some weak assumptions on the statistical properties of the different random sources involved. In particular, the independence of subsequent queuing delays is used as a property that allows to distinguish between large delays caused by queuing (or other related random effects) and large delays caused by a change of network configuration or by change of the master clock.

It has been shown by means of a simulation model that this method leads to a significantly improved performance of the synchronization compared to a synchronization method that does not consider queuing delays. In the simulation model, an accuracy of few ns has been obtained even under high utilization of the network switch.

5. References

[1] G. Bolch, S. Greiner, H. de Meer, K.S. Trivedi: Queueing Networks and Markov Chains. John Wiley & Sons. 1998.

[2] V.G. Kulkarni: Modeling, Analysis, Design and control of Stochastic Systems. Springer, 1999, ISBN 0-387-98725-8

[3] P. J. Rousseeuw, A. M. Leroy: Robust Regression and Outlier Detection", John Wiley & Sons, New York, 1987,2003.

[4] www.imaginethatinc.com

Synchronizing IEEE 1588 clocks under the presence of significant stochastic network delays

Carmelo Iantosca[1], Christoph Heitz[1], Hans Weibel[2]

[1]Institute for Data Analysis and Process Design
[2]Institute of Embedded Systems

Overview

- **Background and aim of present study**
- **Stochastic properties of transit delay**
  - Average transit time
  - Jitter
  - Queuing
- **Empirical results**
- **Robust estimation of transit delay**
- **Results with simulation test bench**

IEEE 1588 in standard ethernet networks

- Aim of present study: Use IEEE 1588 in networks without boundary clocks while retaining high precision
- Problem: How to cope with stochastic network delays by queuing in switches?
- Solution: Exploit stochastic properties of network delay

Variability of delay time

- Transit time may change due to change of network configuration
- Queuing at switches
- Jitter

$$D = D_{net} + D_q + \varepsilon$$

Delay

Average transit time for given network configuration

Queuing delay

jitter

Stochastics of network transit time

$$D = D_{net} + D_q + \varepsilon$$

Constant over many packets. Occasional sudden changes

Always present

Mean value: 0

Variance: small

Often zero (depends on utilization).

If present: Large value, large variability.

Queuing delay

$$D = D_{net} + D_q + \varepsilon$$

Stochastic properties:

- $D_q = B \cdot W$
  B = binary variable with values 1 (prob. ρ) / 0 (prob. 1-ρ),
  ρ = utilization of server
  W = positive random variable: waiting time in queue IF server is busy at arrival
- E(W) is 1…1000 μs
- Sequence B(i) is positively correlated (network congestion may last for longer time periods)
- Sequence W(i) is uncorrelated (status of queue different for each packet)

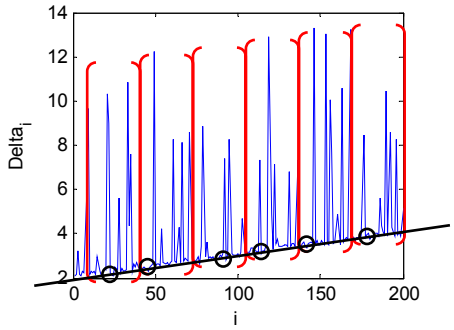
Requirements of data analysis

$$\Delta(i) = \alpha \cdot t_{sent}^{(master)}(i) + S + D_q(i) + \varepsilon(i)$$

- Aim of data analysis:
  - Estimation of drift α
  - Estimation of S
- Required: No sensitivity of estimators to large values of D_q: **Robust estimation**

Robust estimation of drift

- Divide observation window in M equal frames
- For each frame k: determine the *minimum* value of all Δ(i) in this frame
  - y_k is the minimum value
  - x_k is the arrival time of the corresponding packet

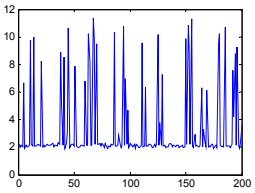


- Estimate drift by linear regression of the data (x_k,y_k), i=1,2,..,M

Robust estimation of S

S = constant part of Δ(i)

Used data: drift compensated time series $\Delta_{corr}(i)$



Idea: Replace usual exponential smoothing

$$\hat{S}(i) = \hat{S}(i-1) + (1-\gamma)\cdot\left(\Delta_{corr}(i) - \hat{S}(i-1)\right)$$

by robust exponential smooting

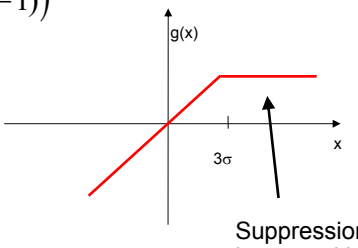
$$\hat{S}(i) = \hat{S}(i-1) + (1-\gamma)\cdot g\left(\Delta_{corr}(i) - \hat{S}(i-1)\right)$$

Weighting function: small deviations are treated as they are, large positive variations are suppressed

Robust estimation of S

$$\hat{S}(i) = \hat{S}(i-1) + (1-\gamma)\cdot g\left(\Delta_{corr}(i) - \hat{S}(i-1)\right)$$

$$g(x) = \begin{cases} x, & x < 3\sigma \\ 3\sigma, & x \geq 3\sigma \end{cases}$$



Suppression of large positive Δ(i)

- Queuing delays are suppressed to low values (not zero)
- Required information: Variance σ^2 of jitter (only order of magnitude)
- Computationally cheap, few storage space required

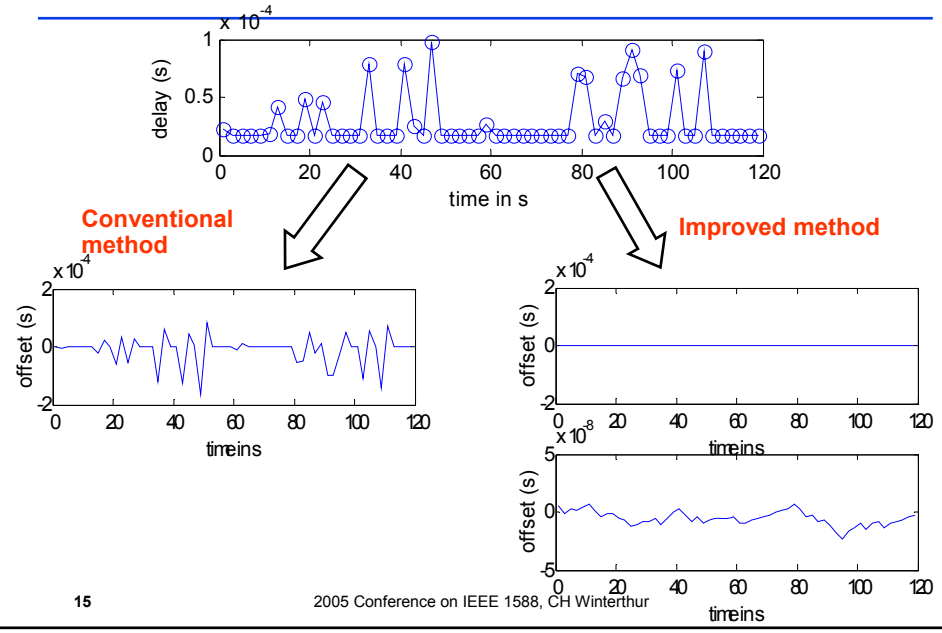
Leap detection

- Leap detection necessary for sudden change of network configuration (change of D_{net})
- Problem: Δ_{corr} changes strongly, but not due to queuing. How to avoid that this deviation is damped out?
- Idea:
  - After change of D_{net}: subsequent $\Delta_{corr}(i)$ differ by ~σ.
  - If queuing occurs: subsequent $\Delta_{corr}(i)$ differ by >> σ
- Construction of detector whose statistical properties can be calculated
- For moderate utilization, leaps are detected quickly. For higher utilization, leap detection takes more time

Test of new algorithm

- **Simulation based test bench**
- **Comparison:**
  - Typical PTP implementation (networks with boundary clocks): can deal with small variations (jitter) and large leaps (change of master clock or topology)
  - Improved synchronization method

Results



Conclusions

- New method of synchronizing IEEE 1588 clocks has been developed
- Able to cope with random queuing delays
- Method based on stochastic properties of different sources of transit delay
  - Network configuration
  - Jitter
  - queuing
- Algorithms: robust statistical estimation methods
- First results very promising

Thank you for your attention!

Average transit delay

$$D = D_{net} + D_q + \varepsilon$$

D_{net} = average transit delay (jitter = 0, no queueing)

Stochastic properties:

- $D_{net}(i)=D_{net}(i-1)$ with probability p ($p \approx 1$)
- $D_{net}(i) \neq D_{net}(i-1)$ with probability 1-p

Example: One change of network configuration per day, Sync messages all two seconds (i.e. 43200/day)

→ $p = 1 - 2.3 \cdot 10^{-5}$

Jitter

$$D = D_{net} + D_q + \varepsilon$$

Stochastic properties:

- $E(\varepsilon) = 0$
- Not necessarily normal distributed
- Variance: σ^2
- Typical value of σ: some hundreds of nanoseconds up to 1 μs

Precise Timing in a Residential Ethernet Environment

Michael Johas Teener - Broadcom
mikejt@broadcom.com
Felix Feng - Samsung
feng.fei@samsung.com
Eric Hyunsurk Ryu - Samsung
eric_ryu@samsung.com

BROADCOM. connecting everything

Agenda

- Introduction to Residential Ethernet
- Timing synchronization in ResE
- Differences between IEEE 1588 and ResE
- Using ResE as an IEEE 1588 subnet

Introduction to Residential Ethernet

BROADCOM. connecting everything

What is Residential Ethernet?

- Simple enhancement to IEEE 802.1 bridges to support streaming QoS
  - 2 ms guaranteed latency through 7 bridges
  - Admission controls (reservations) for guaranteed bandwidth
  - Precise timing and synchronization services for timestamps and media coordination
    - May require extra timing service from 802.3 MAC
- Trade group to provide trademark "enforcement" of otherwise optional features
  - Require useful bridge performance, network management, PoE management, auto-configuration features

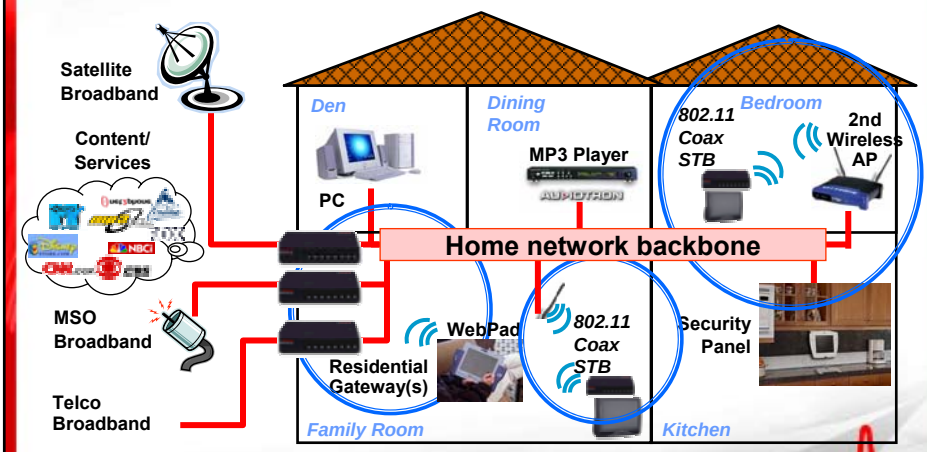
Why is it needed? (1)

- Common IT-oriented networks have inadequate QoS controls
- All use 802.1 “priorities” (actually, “traffic class”)
  - Ethernet is the best
    - … but it’s easy for the customer to misconfigure or overload
    - … no guarantees
  - Wireless has inadequate bandwidth and excessive delays for whole-home coverage
    - … 802.11n and UWB work for non-critical applications, or short range
    - … no guarantees
    - … and the backbone for the wireless attachment points is? ▪

Why is it needed? (2)

- Proposed CE-based networks need new media or are expensive
  - MoCA requires coax everywhere, and is not cheap, and does not carry power, and has modest performance
    - … but it’s part of the solution
  - Power line is not cheap, has modest performance, is susceptible to interference, and is blocked by protection circuits
    - … but it’s part of the solution
  - 1394b/c long distance requires optical fiber or CAT-5, is not cheap
    - … but even this is part of the solution

Digital Home Media Distribution



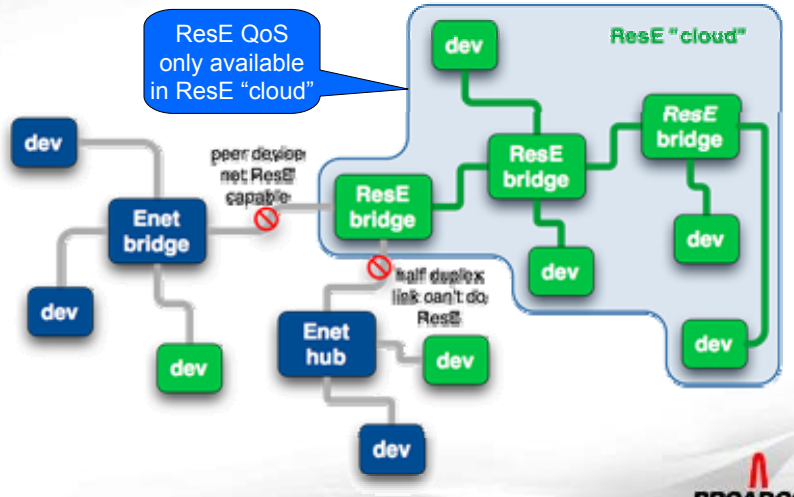
Where will Residential Ethernet be used?

- Backbone for home
  - Highest quality/lowest cost way to interconnect wireless A/Ps
  - "Perfect" QoS, requires the least customer interaction
- Within the entertainment cluster
  - Trivial wiring, no configuration, guaranteed 100/1G/2.5G *per device*, not just *per room*
  - PoE for speakers, extra storage (HD/optical), wireless A/Ps, other lower-power devices
  - Ideal long-term replacement for 1394
- Numerous non-"residential" applications
  - Professional audio/video studios
  - Industrial automation
  - Test and measurement

Proposed architecture

- Propose changes to both IEEE 802.3 (Ethernet) and IEEE 802.1Q (bridges/switches)
- Three basic additions to 802.3/802.1
  - Traffic shaping and prioritizing,
  - Admission controls, and
  - Precise synchronization

Topology & connectivity



Traffic Shaping and Priorties

- Endpoints of ResE network must “shape traffic”
  - Schedule transmissions of streaming data to prevent bunching, which causes overloading of network resources (mainly switch buffers)
  - Probably limit by “x bytes in 125usec” and “x bytes in 2ms” depending on traffic class
- Mapping between traffic class and priorities

Traffic Class?

- 802.1p introduced 8 different traffic classes
  - Usually implemented as strict priorities
  - Highest (7 & 8) reserved for network management (low utilization)
  - Next two for streaming (5 & 6)
  - Lowest four for “best effort”
- Proposal:
  - Class 6 is for lowest latency streaming
    - Roughly 125usec per bridge hop: interactive audio/video
  - Class 5 is for moderate latency streaming
    - Perhaps 2ms per bridge hop: voice over IP, movies

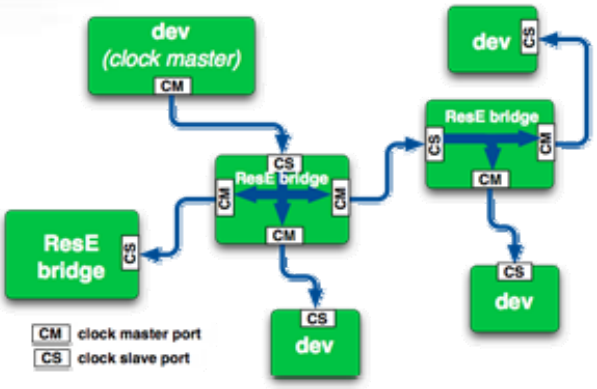
Admission controls

- Streaming priority mechanism can reliably deliver data with a deterministic low latency and low jitter
  - but only if the network resources (bandwidth, in particular) are available along the entire path from the talker to the listener(s).
- For ResE it is the listener's responsibility to guarantee the path is available and to reserve the resources.
- Done via a new 802.1ak "Multiple Registration Protocol" application: SRP ("Simple Registration Protocol")
  - Registers streams as multicast address/bandwidth needed pairs

Precise synchronization

- ResE devices will periodically exchange timing information
  - both devices synchronize their time-of-day clock very precisely.
- This precise synchronization has two purposes:
  - to enable streaming traffic shaping and
  - provide a common time base for sampling data streams at a source device and presenting those streams at the destination device with the same relative timing.

Network master clock



- There is a single device within a ResE "cloud" that provides a master timing signal.
  - All other devices synchronize their clocks with this master.

Master clock selection

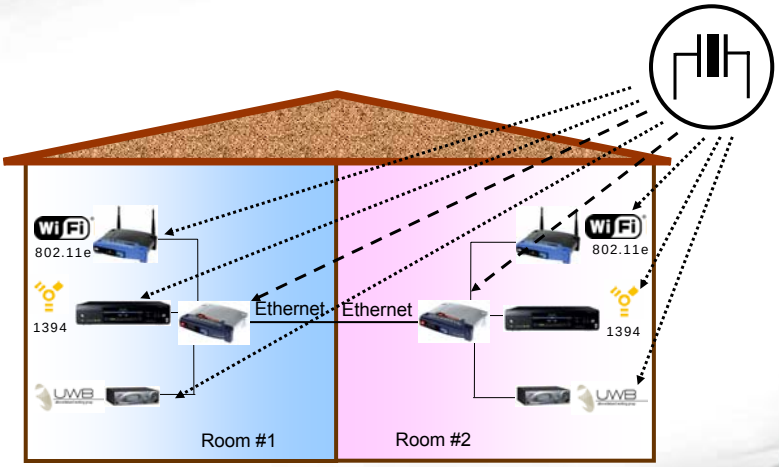
- Selection of the master is largely arbitrary (all ResE devices will be master-capable), but can be overridden if the network is used in an environment that already has a "house clock".
  - Professional A/V studios
  - Homes with provider 1588 service

Changes needed in existing products

- Endpoint device needs
  - Timer
  - Streaming traffic transmit FIFO(s)
    - (streaming receive use existing FIFO)
  - Best to have dedicated ports for streaming data
    - MPEG-TS, I^2S, etc., like existing 1394 links
- Bridges
  - ResE MACs
  - Streaming routing/filtering
    - similar to asynch logic
  - Admission control firmware
    - similar to 802.1 multicast and VLAN management
  - Timing propagation

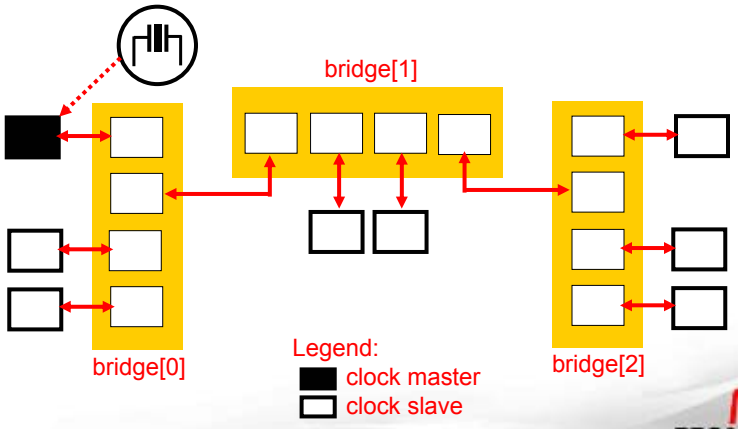
Timing Synchronization in ResE

House reference clock



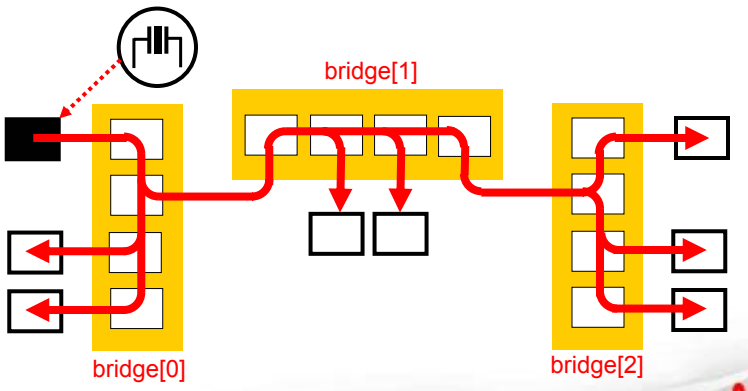
Cascaded TOD synchronization

Physical topology constraints



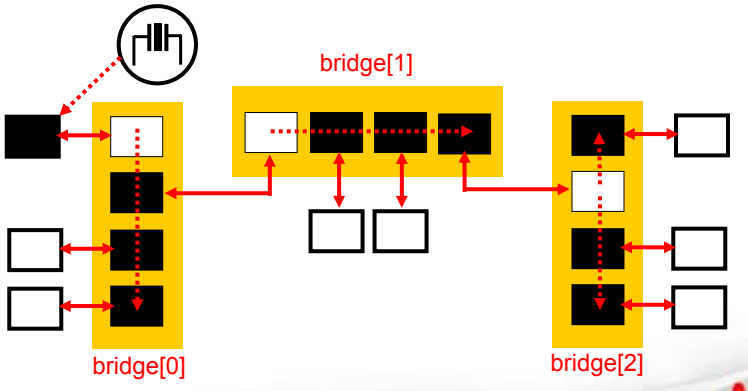
Cascaded TOD synchronization

Wall-clock distribution model



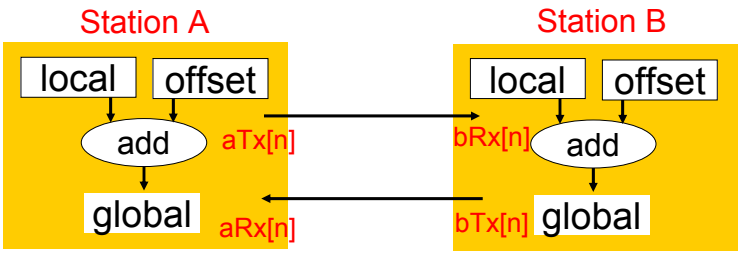
Cascaded TOD synchronization

Cascaded adjacent-synchronization hierarchy



Adjacent-station synchronization

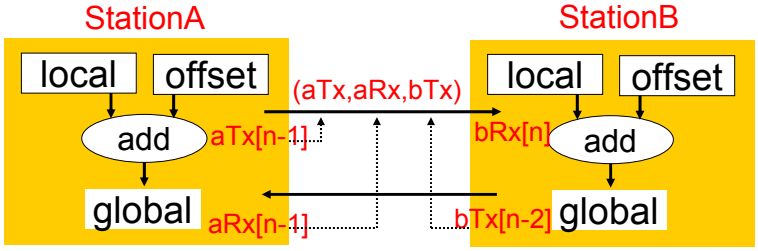
Timing snapshots



Adjacent-station synchronization

Snapshot value distribution

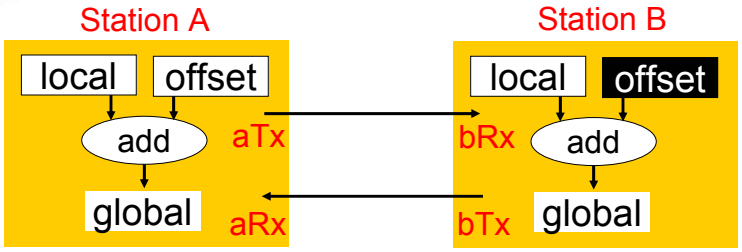
(information for stationB is time A sent previous snapshot, time A received B's previous snapshot, and time B sent snapshot before that)



Transmit timings are always for previous snapshot because they are recorded when the snapshot was sent, and are not available while the packet is in the process of being sent

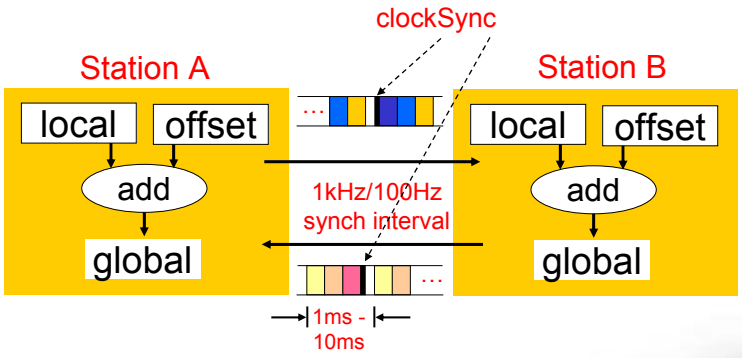
Adjacent-station synchronization

StationB offset adjustments



- rxDelta = (bRx[n-1] – aTx[n-1]);
- txDelta = (bTx[n-1] – aRx[n-1]);
- clockDelta = (rxDelta – txDelta) / 2;
- cableDelay = (rxDelta + txDelta) / 2;
- offsetB = offsetA – clockDelta;

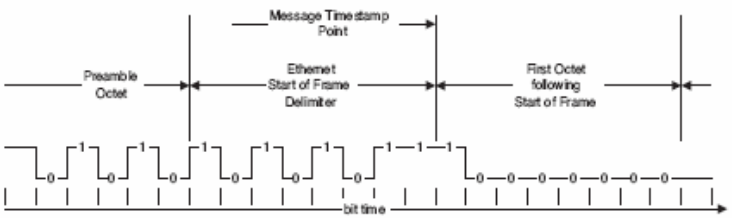
Adjacent station synchronization



Timing specifics...

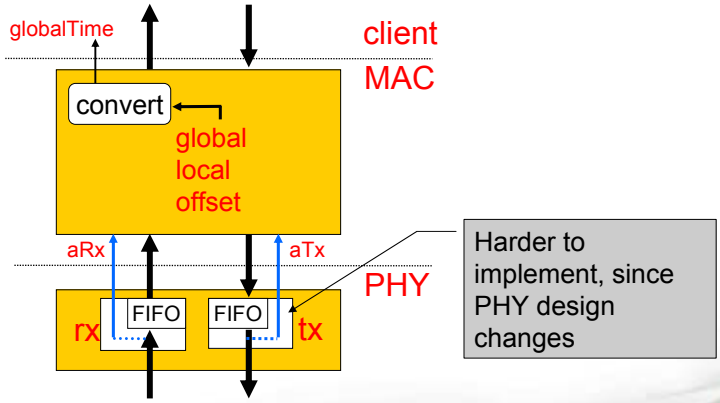
- Could add to 802.3 PHY specs:

(from IEEE 1588-2002, subclause D.1.1, page 127)

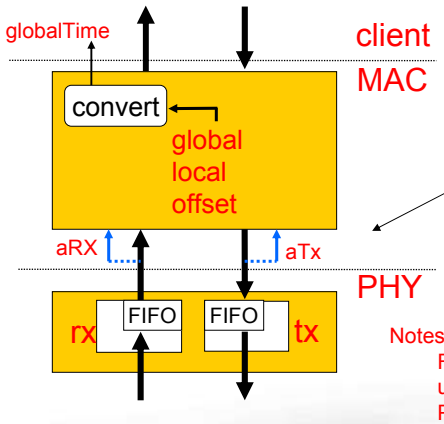


- But realistically, more likely to get "when first data symbol of frame is transmitted to or received from PHY"
  - Less precise ... but ResE has frequent clock updates

A PHY-based design model



A MAC-based design model



Easier to implement, since no change to PHY

Notes:
FIFOs add uncertainty, but PHY buffers are small (0-32 bits)

Differences Between ResE and 1588

Differences: No Options

- ResE must have “consumer-friendly” cost structure
  - 5 port 100baseT switches sell for US$30
  - Needs to use low cost time reference (standard crystal, much less than US$1)
- No IT manager in the home
  - Must be really self-configuring
  - Use UPnP or similar management

Differences: Two-way Only

- Scaling and cost structure dictate requiring just one method
- Two-way only for all purposes is simplest
- OK because only 100baseT and better will be used
  - And small packets (everything will fit in Ethernet-minimum 64 bytes)
  - And frequent updates (1ms - 10ms)

Differences: Direct Layer 2 Point-to-point

- ResE runs only on full duplex links
- No intermediate devices between participating nodes
  - No hubs, no non-participating switches, no routers
- Frame transport delay is tightly bound
  - Media and PHY coding/decoding are only uncertainties

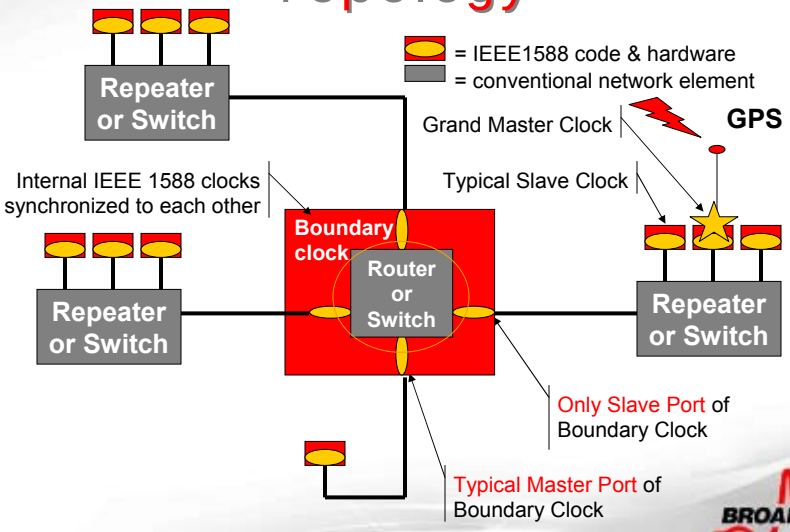
Differences: Single Method for Clock Cascading

- All ResE switches are similar to boundary clocks
- Only a single method to be used for synchronizing master/slave clocks within switch
  - Garner/Hollander presentation will outline proposed methods and performance analysis

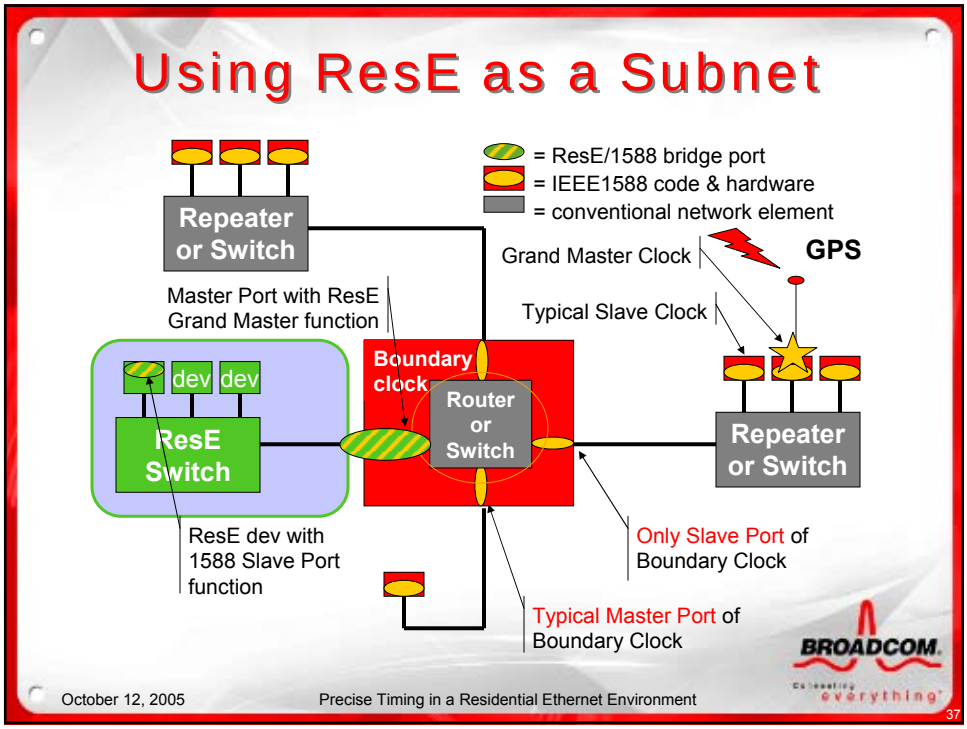
Using ResE as an IEEE 1588 Subnet

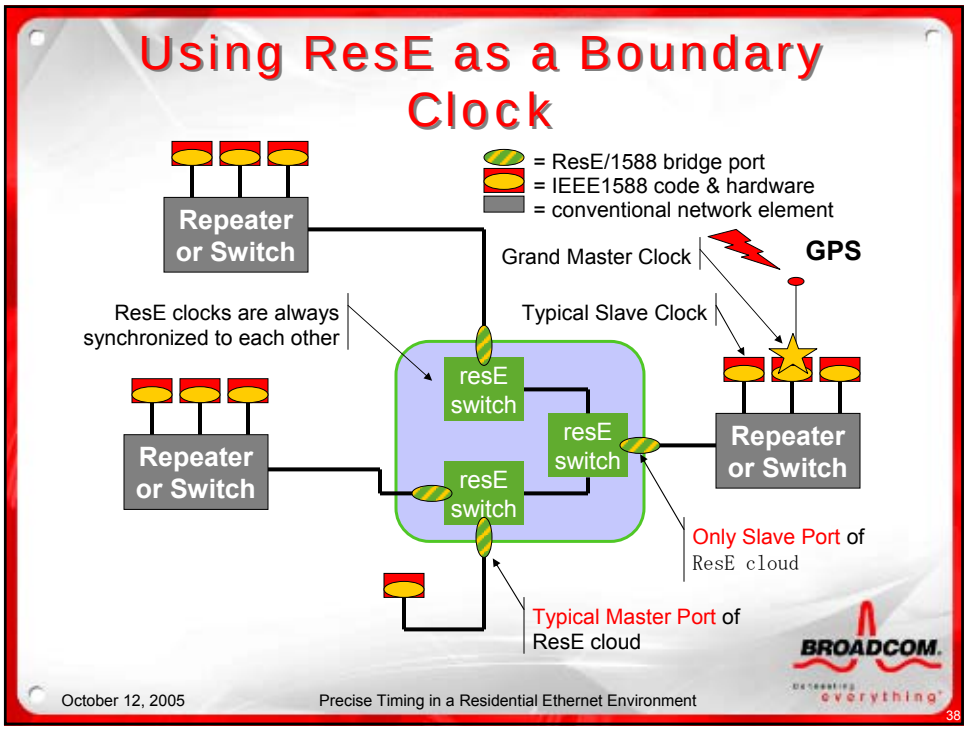
BROADCOM. connecting everything

IEEE 1588 Multiple Subnet Topology



BROADCOM. connecting everything


Using ResE as a Subnet
= ResE/1588 bridge port
= IEEE1588 code & hardware
= conventional network element
Repeater or Switch
Grand Master Clock
GPS
Master Port with ResE Grand Master function
Typical Slave Clock
Boundary clock
Router or Switch
dev
dev
ResE Switch
Repeater or Switch
ResE dev with 1588 Slave Port function
Only Slave Port of Boundary Clock
Typical Master Port of Boundary Clock
BROADCOM
October 12, 2005
Precise Timing in a Residential Ethernet Environment
37


Using ResE as a Boundary Clock
= ResE/1588 bridge port
= IEEE1588 code & hardware
= conventional network element
Repeater or Switch
Grand Master Clock
GPS
ResE clocks are always synchronized to each other
Typical Slave Clock
resE switch
resE switch
resE switch
Repeater or Switch
Repeater or Switch
Only Slave Port of ResE cloud
Typical Master Port of ResE cloud
BROADCOM
October 12, 2005
Precise Timing in a Residential Ethernet Environment
38

Conclusions

- Residential Ethernet represents another building block in timing-aware systems
- ResE can be used as an element in a 1588 architecture
  - Providing the performance is adequate
  - Can provide either boundary clock or subnet functionality

Analysis of Clock Synchronization Approaches for Residential Ethernet

Geoffrey M. Garner (Consultant) Kees den Hollander
SAIT, Samsung Electronics
gmgarner@comcast.net, denhollander.c.j@samsung.com

Abstract

Residential Ethernet (ResE) is a new standardization activity in IEEE 802 that is considering extensions to Ethernet to allow the transport of time-sensitive traffic. Applications that ResE is expected to carry will include digital video, high-fidelity digital audio, and gaming traffic, as well as traditional non-time-sensitive traffic (e.g., data traffic). One goal of ResE is to allow a single network infrastructure in the residence to carry both time-sensitive and non-time-sensitive applications.

The Audio/Video (A/V) applications for ResE have tight jitter and wander requirements; in addition, applications where A/V content is delivered to multiple locations (e.g., gaming, digital stereo speakers) may have tight time synchronization requirements. To meet the jitter, wander, and time synchronization requirements for the applications, time synchronization must be provided to the ResE endpoints. Several approaches for providing time synchronization, all of which are based on principles used in IEEE 1588 and employ time stamps, are being considered. However, a number of algorithms for using the time stamp information have been suggested, e.g., instantaneous phase adjustments, instantaneous phase adjustments and less frequent instantaneous frequency adjustments, phase adjustments and possibly less frequent frequency adjustments with filtering, phase and frequency adjustments using phase-locked loops (PLLs), and use of transparent clocks. The jitter and wander performance and delivered time accuracy of each scheme will depend on the rate of phase and, if appropriate, frequency adjustments (which is limited by the rate at which time stamp information is transported), bandwidth and gain-peaking of the various filters and/or PLLs, quality of the node clocks (frequency tolerance, phase noise level and characteristics, etc.), and size of the network.

This paper develops simulation models to analyze several of the approaches (the remaining approaches will be considered in future work). Simulation results are obtained and compared with each other and with end-to-end application jitter and wander requirements. While this is initial work, the eventual goal is to provide input for deciding which approach is best for Residential Ethernet.

1. Introduction

Residential Ethernet (ResE) is a new standardization activity in IEEE 802 that is considering extensions to Ethernet to allow the transport of time-sensitive traffic. Applications that ResE is expected to carry will include digital video, high-fidelity digital audio, and gaming traffic, as well as traditional non-time-sensitive traffic (e.g., data traffic). One goal of ResE is to allow a single network infrastructure in the residence to carry both time-sensitive and non-time-sensitive applications. Ethernet is a ubiquitous and inexpensive network technology, and is ideally suited for this purpose.

Two key ResE requirements are (1) guaranteed QoS for time-sensitive applications, and (2) minimal administration by users. Guaranteed QoS means that application jitter, wander, time synchronization, and latency requirements are met. Minimal administration means that administration and provisioning should not be actively required on an ongoing basis. To meet these requirements, ResE will need features that include bandwidth reservation, admission control, and network synchronization. ResE also will use the priority class mechanism of current Ethernet. Finally, ResE rates will initially include 100 Mbit/s and 1 Gbit/s, with 10 Gbit/s added in the future.

The Audio/Video (A/V) applications for ResE have tight jitter and wander requirements. For example, compressed digital video delivered in the form of MPEG-2 packets has a peak-to-peak phase variation requirement that can range from 1 μs for the case where the video originates in the residence to some fraction of 50 μs for the case where the video is delivered by a service provider (the precise fraction depends on the jitter and wander budget allocations).

Digital audio has a peak-to-peak jitter requirement of 10 ns, measured with a 200 Hz measurement filter for consumer-grade interfaces. These jitter and wander requirements for A/V applications must be met by the A/V signals that are demapped from ResE packets and delivered to the respective codecs. In addition, applications where A/V content is delivered to multiple locations (e.g., gaming, digital stereo speakers) may have tight time synchronization requirements. To meet the jitter, wander, and time synchronization requirements for the applications, time synchronization must be provided to the ResE endpoints.

Several approaches are being considered for providing time synchronization, all of which are based on principles used in IEEE 1588 Precision Time Protocol (PTP) [1] and Network Time Protocol (NTP) [2], and employ time stamps. In most cases, the time-stamps are two-way and therefore analogous to the Sync and Delay_Req messages of IEEE 1588. However, a number of algorithms for using the time stamp information have been suggested, e.g., instantaneous phase adjustments, phase adjustments with filtering, instantaneous phase adjustments and less frequent instantaneous frequency adjustments, phase adjustments and less frequent frequency adjustments with filtering, phase and frequency adjustments using phase-locked loops (PLLs), and use of transparent clocks [3]. The jitter and wander performance and delivered time accuracy of each scheme will depend on the rate of phase and, if appropriate, frequency adjustments (which is limited by the rate at which time stamp information is transported), bandwidth and gain-peaking of the various filters and/or PLLs, quality of the node clocks (frequency tolerance, phase noise level and characteristics, etc.), and size of the network.

This paper develops a simulation model for the approaches based on instantaneous phase adjustments and instantaneous phase adjustments with less frequent instantaneous frequency adjustments, both with and without filtering. Simulation results are obtained for several cases that include the effects of node clock phase noise and time measurement granularity. The results are compared with the A/V application jitter and wander requirements. The present work is initial work; more detailed analysis of cases that cover ranges of parameters is planned for future work. The modeling and simulation of the other approaches (e.g., PLL filtering, transparent clocks) is also an area for future work.

The paper is organized as follows. Section 2 summarizes the jitter and wander requirements for A/V applications that are expected to be carried by ResE, and presents an example Hypothetical Reference Model (HRM) for the transport of compressed digital video. Section 3 describes the fundamental time stamp exchange process and summarizes the various approaches/schemes for using the time stamp information. Section 4 develops the simulation model for the three approaches considered here. Section 5 presents the simulation cases and results. Section 6 presents conclusions and indicates future work. The clock phase noise model used in the simulations is described in the Appendix.

2. Jitter and wander requirements for Residential Ethernet time-sensitive applications

Residential Ethernet will carry compressed digital video in the form of MPEG-2 or MPEG-4 and high-fidelity (e.g., CD quality) digital audio. In the future, ResE may carry uncompressed digital video, but this will likely require the use of 10 Gbit/s Ethernet to be practical. Note that MPEG-2 and MPEG-4 are not basic video formats; rather, they are formats for compressing and transporting uncompressed digital video that was originally produced by digitizing analog video. Detailed descriptions of digital video and digital audio applications are given in [4] and [5], respectively, and the jitter and wander requirements for these applications are discussed in [6]. The jitter and wander requirements are summarized below.

2.1. Video applications

Two classes of uncompressed digital encodings have been standardized: (1) direct sampling of a composite video signal, and (2) individual sampling of each component of a component video signal. Sampled composite NTSC and PAL signals with nominal rates of 143 Mbit/s and 177 Mbit/s, respectively, are defined in [7]. Sampled component video signals with nominal rates of 270 Mbit/s and 360 Mbit/s are defined in [7] and [8]. These 4 signals are standard definition TV (SDTV). Sampled High Definition TV (HDTV) component video signals with nominal rates of 1.485 Gbit/s and 1.485/1.001 Gbit/s are defined in [9]. Jitter, frequency offset, and frequency drift requirements for SDTV and HDTV signals are summarized in columns 2 and 3 of Table 1 [7], [9].

Compressed digital video is the initial focus for ResE because the uncompressed digital video rates are appreciable relative to or exceed the 100 Mbit/s and 1 Gbit/s Ethernet rates and the capacities of transport networks that deliver video to the residence. The MPEG-2 standard defines a scheme for compressing and packetizing digital video and audio into Packetized Elementary Streams (PESs) [10]. The PESs for a single program may be multiplexed into a Program Stream (PS), with multiple PES packets combined into larger PS packs. Alternatively, the PESs from multiple programs may be multiplexed into a Transport Stream (TS). In this case, each PES packet is mapped into one or more 188 byte TS packets (in general, the TS packet

is smaller than the PES packet). A PS contains a single program, and is therefore traceable to a single clock; it is typically used in a DVD application. A TS contains multiple programs that are traceable to different clocks and have been multiplexed. The TS multiplexing process includes a rate adjustment of the individual PESs to a TS Reference Clock.

The MPEG-2 packs or packets are mapped into Ethernet frames at the ResE ingress, transported over ResE, and demapped at the egress. However, the full reference model may also involve transport over one or more service provider networks (see Figure 1). The uncompressed digital video is encoded in MPEG-2 at the source and transported over several service provider networks. The PES and TS are created at Ref. Points B and C, respectively; note that the TS may contain additional PESs. The TS packets enter the residence, and an interworking function demaps them from the service provider transport protocol and maps them into Ethernet frames. The frames traverse the ResE, and the TS packets are demapped at Ref. Point D. The PES is then demultiplexed at Ref. Point E and input to the MPEG-2 decoder.

The accumulated jitter and wander at the ResE demapper (Ref. Point D) must be within the MPEG-2 decoder requirements, which are summarized in columns 4 and 5 of Table 1 [10]. Note that there are no jitter requirements; rather, there are requirements on peak-to-peak phase variation without any high-pass measurement filter. There actually are two requirements, one for the case where the MPEG-2 video is transported across a network, and the other for the case where there is no network transport. The former requirement applies for the case of transport over ResE and/or a service provider network to the residence (Ref. Point D and E). The latter requirement applies to Ref. Point B and C, whose jitter and wander are the same as that for Ref. Point E and D, respectively, if no networks are present.

Figure 1 also illustrates the use of ResE application time stamps to control the jitter and wander of the TS packets at the ResE egress. These time stamps are measured relative to the ResE synchronized network clocks. A time stamp is created for each TS packet when it is mapped into an Ethernet frame, relative to the network clock at the ingress; the time stamp is then used to determine when to deliver the TS packet to the MPEG-2 TS demux (Ref. Point D) using the synchronized network clock at the egress. Note that these time stamps are part of the MPEG-2 (or MPEG-4) TS to Ethernet adaptation function; they are distinct from the time stamps used within MPEG-2 or MPEG-4, and also from the IEEE 1588-like time stamps used by the ResE node clocks to for network synchronization.

2.2 High-fidelity audio applications

The primary audio applications that ResE will initially transport include digital audio that originates in the residence on a CD and analog audio that originates in the residence and is digitized prior to transport. The digital audio rate is controlled by the source clock or sampling clock. The digital audio signal is then transported to a receiver, where clock and data recovery are performed. The recovered clock is then filtered further to produce a sampling clock for D/A conversion. The additional filtering is needed because the jitter requirements for the sampling clock to produce high-fidelity audio are much more stringent than the requirements for data recovery.

The interface between the transmitter and receiver is standardized separately for consumer and professional applications [11]. The data is carried in 64-bit (128 Unit Interval (UI), with 2 UI/bit line coding) frames. The nominal basic frame rates are 44.1 kHz for consumer applications and 48 kHz for professional applications. In addition, double, quadruple, half, and quarter rates are defined. The highest rate is four times the 48 kHz rate, which corresponds to 24.576 Mbit/s (128 UI/frame).

In carrying digital audio over ResE, the audio frames are mapped into Ethernet frames at the transmitter, transported over ResE, and demapped at the receiver. The accumulated jitter and wander at the demapper must be within the requirements of the additional filter that produces the sampling clock for the D/A conversion. The requirements are summarized in columns 6 and 7 of Table 1 [11].

2.3 End-to-end application jitter and wander requirements expressed in terms of Maximum Time Interval Error

The requirements of Table 1 can be expressed conveniently in terms of Maximum Time Interval Error (MTIE). MTIE is peak-to-peak phase variation for an observation interval, expressed as a function of the interval length. The peak-to-peak is taken over all possible observation intervals of the given length in the measurement sample. The rigorous mathematical definition of MTIE is given in [12], along with the following expression that can be used to compute an estimate of MTIE from measured or simulated data

$$\mathrm{MTIE}(n\tau_0) = \max_{1\le k\le N-n}\left(\max_{k\le i\le k+n} x(i) - \min_{k\le i\le k+n} x(i)\right). \quad (1)$$

$$n = 1, 2, \ldots, N-1$$

Here, $x(i)$ is the ith phase offset sample (out of N total samples), τ_0 is the sampling time, $n\tau_0$ is the observation interval, and $N\tau_0$ is the measurement interval.

MTIE requirements equivalent to the jitter and wander requirements of Table 1 are shown in Figure 2. MTIE for each A/V signal, at the input to the codec at the ResE egress, must not exceed the respective curve.

Detailed derivations of the masks are given in [6]. The requirement for MPEG-2 transported over networks (50 μs phase variation over approximately 600 s) is least stringent. The digital audio and uncompressed digital video jitter requirements are much more stringent (10 ns and less than 1 ns, respectively). The masks are network limits; for the case where the application is transported to the residence via service provider networks, ResE gets only an allocation of the total limit, with the remainder going to the service providers.

3. Approaches for providing ResE Synchronization

Several approaches, based on the exchange of time stamps between a master and slave clock, are being considered for ResE synchronization. The same principles are used in IEEE 1588 [1] and NTP [2]. It is envisioned that all the clocks in the ResE will be synchronized by one clock, termed the Grand Master. A slave clock will synchronize to a master which, in turn, will synchronize to its master; the synchronization chain will continue to the Grand Master.

The basic procedure for computing a correction to a slave clock is shown in Figure 3. The master and slave clocks are free-running with frequencies f_0 and f_1, respectively, and also initially indicate different times. In the kth message exchange, the slave initially sends a message to the master that contains the time $T^S_{1,k}$ that the message is sent. The master notes the time it receives this message, $T^M_{2,k}$ and, at a later time, sends a message back to the slave containing the time the slave sent the first message, the time the master received that message, and the time $T^M_{3,k}$ that the master sends the current message. The slave notes the time it receives this message, $T^S_{4,k}$. Under the assumption that the propagation delays for the two messages are the same, the slave computes a correction u_k

$$u_k = \frac{\left(T^M_{2,k} - T^S_{1,k}\right) - \left(T^S_{4,k} - T^M_{3,k}\right)}{2}. \qquad (2)$$

This correction, if added to the current slave clock time, will synchronize the slave to the master clock. The error in the synchronization is of the order of the frequency difference between the master and slave multiplied by the duration of the message exchange.

[1] and [2] do not specify how the corrections should be used; a number of approaches have been suggested [17], and are summarized in Table 2. Since the master and slave clocks are free running with different frequencies, the message exchange must be repeated over time. In the simplest approach, the slave time is adjusted instantaneously at each message exchange. However, this adjustment results in an instantaneous phase step, which could result in exceeding some or all of the application jitter/wander requirements. Therefore, it is assumed that some form of filtering is performed at the network egress in all the approaches. The filtering may take the form of a digital filter running at the local egress free-running clock rate, or may be a full PLL whose frequency is adjusted based on filtered phase corrections.

The time stamp exchange described above is a two-way exchange; the master clock always responds to a message from the slave. However, the spatial extent of a typical ResE network is expected to be on the scale of a residence, with propagation delays stable and small (on the order of several μs) compared to the inter-message time (no smaller than 1 ms and likely an order of magnitude or more larger). Therefore, it is not necessary to always use a two-way time stamp exchange; instead, one-way time stamps could be used with less frequent two-way exchanges to obtain propagation delay as is done in IEEE 1588 [1] (Approach 1 in Table 2). The most recently computed propagation delay would be used by the slave when a one-way time stamp is received to compute the correction.

Approaches 2 – 5 in Table 2 concern the use of the clock corrections u_k, whether they are computed via one-way or two-way time stamps. First, the entire u_k value can be added to the current free-running slave clock time immediately to obtain the corrected time value, at each successive slave clock node (Approach 2). Second, the change in slave clock phase and master clock phase over some number of message intervals can be used to compute the frequency offset of the slave relative to the master (Approach 3). This offset can be used to obtain an improved estimate of slave clock phase, i.e., the phase the slave clock would have if it ran at the current estimate of the master clock rate. This improved slave clock phase estimate can then be used to obtain the slave clock correction u_k, which should be much smaller than the value that would be obtained if the frequency correction were not made. As in Approach 2, the u_k is added instantaneously to obtain corrected slave clock time. In Approach 4, the sequence of slave clock corrections u_k are filtered with a digital filter running at the local clock rate, and the filtered stream of corrections are added to the free running clock time. This has the effect of smoothing the instantaneous phase jumps that occur when the u_k are added directly. Note that whether or not filtering is done, the corrections must be accumulated on traversing the chain of slave clocks, because each u_k gives the correction needed to synchronize the slave to its upstream master, but what is actually desired is to synchronize the slave to the grandmaster at the beginning of the chain. If filtering is performed at the local clock rate and the filters at all nodes have the same properties (bandwidth, gain peaking, etc.), then the result is the same whether the

unfiltered corrections are accumulated and filtered at the egress or the filtered corrections are obtained at each node and accumulated, except for small differences due to the filters at the different nodes running at slightly different rates. This is because the filters are linear and run at nominally the same rate, and the u_k depend on unfiltered quantities. Finally, in Approach 5 the u_k are used in a PLL implementation to adjust the slave clock rate; this means that a particular u_k will affect future message times and future u_j ($j > k$). In this approach, placing a PLL at each intermediate node versus having a single PLL at the egress node will give different results.

Approach 5 results in a chain of PLLs. Such a scheme can produce acceptable jitter and wander accumulation with suitable PLL bandwidth, gain peaking, and noise generation (it is well-known that excessive gain peaking can result in large phase accumulation). However, depending on the actual requirements, this may result in the need for expensive oscillators. Conversely, it is not yet clear what the relative performance of Approaches 2 – 4 is relative to Approach 5. While the analysis of the present paper is limited to Approaches 2 – 4, all the approaches will be analyzed before a decision is made on which approach is most suitable for ResE.

Another scheme has been proposed [3] to reduce the need for long chains of PLLs. If synchronization is not needed at an intermediate node, it is possible to relay a message that traverses the node and measure the time the message resides in the node. This residence time may be variable, but as long as it is measured and relayed with the message, the eventual slave clock that receives the message can correct for this time. Such an intermediate node is termed a "transparent clock" [3]. If the message traverses several transparent clock nodes, the residence time can be accumulated. Since the transparent clock is free running, the correction will be in error by an amount equal to the frequency offset of the transparent clock multiplied by the residence time of the message in the node. It is possible to use Approaches 2 – 5 to adjust the phase and frequency of a transparent clock, but the same considerations indicated for slave clocks apply.

Depending on how the time stamp measurements are made (i.e., the measurements of the $T^S_{1,k}$, $T^M_{2,k}$, $T^M_{3,k}$, and $T^S_{4,k}$.), it may be much easier to make an accurate measurement if the value does not need to be sent in the current time stamp. In Approach 7, the time stamps reflect times delayed by some number of message exchanges. Finally, in Approach 8 the time stamps contain corrected slave clock times rather than the free-running, local clock values (Approach 8 is a modification of Approaches 2 – 4). This would avoid the need to accumulate the corrections, because the time stamps contain times traceable to the grandmaster.

The following sections develop a simulation model for Approaches 2 – 4 and consider several initial simulation cases. As indicated above, the other approaches will be analyzed in future work.

4. Simulation model for synchronization using Approaches 2 – 4

Figure 4 shows a chain of M+1 nodes, indexed from 0 to M. Node 0 is the grandmaster, and node i is the master of node i+1. The successive nodes exchange timestamp messages with each other, as indicated in Figure 3. Each node is timed by a clock that is free-running and has a level of phase noise. Let $y_b^i(t)$ and $n^i(t)$ be the fractional frequency offset expressed as a pure fraction and phase noise process (a random process) expressed in units of time, respectively, for the node i clock. Let $x_b^i(t)$ be the phase offset of clock i relative to UTC (ns). Then, assuming the frequency offset is constant over time and the phase offsets are zero at $t = 0$

$$x_b^i(i) = \int_0^t y_b^i(t)\,dt + n^i(t) = y_b^i t + n^i(t). \qquad (3)$$

We are primarily interested in the synchronization performance relative to the grandmaster, therefore, we can set $y_b^0(t) = n^0(t) = 0$.

Figure 5 shows the details of 3 successive message exchanges (exchanges k, k+1, and k+2) between node i+1 (slave) and node i (master). Let $T_{1,k}$' be the time the kth message (i.e., part of the kth message exchange) from the slave to the master leaves the slave, measured relative to the slave's free-running clock. Let $T_{1,k}$ be this same time measured relative to the master's free-running clock. The times $T_{j,k}$ and $T_{j,k}$', for j = 2, 3, and 4, are defined similarly in Figure 5, with the primed quantities measured relative to the slave's free-running clock and the unprimed quantities measured relative to the master's free-running clock. In addition, let T_m be the nominal time between successive messages from master to slave, D the propagation delay between master and slave (it is assumed that the propagation delay is the same in both directions), and x the time offset between the receipt of the message from the slave by the master and the sending of the response message by the master to the slave. T_m, D, and x are expressed relative to UTC.

As indicated earlier, the propagation delay D is on the order of several μs for a residential network, and T_m will not be less than 1 ms. Then we may assume that $D << T_m$. The master to slave and slave to master message times are, in general, not synchronized, and x ranges from 0 to T_m. Therefore, the probability that messages from the master to slave and slave to master overlap in time is small, and is neglected here.

For each master/slave exchange, the quantity x will be initialized randomly. In addition, depending on whether the master and slave tie their message rates to their local free-running clocks or to their corrected times traceable to the grandmaster, x will vary with

time or remain constant, respectively. In the former case, the change in x over one inter-message time T_m is equal to the frequency offset between master and slave multiplied by T_m, i.e., $(y_b^{i+1} - y_b^i)T_m$. Depending on the relative signs and magnitudes of the frequency offsets, the change in x over T_m can be positive or negative and, over many inter-message times, x will reach T_m or 0, respectively. When this happens, a master to slave message "walks past" the next or previous slave to master message, and x jumps to 0 or T_m, respectively. It will be seen in the simulation results that this impacts the performance for the case where frequency adjustments are not made.

Let $x_{b,k}^i = x_b^i(kT_m)$, i.e., the phase offset of the free-running clock at node i at time kT_m, and $n_k^i = n^i(kT_m)$, i.e. the phase noise of the clock at node i at time kT_m. Then

$$x_{b,k}^i = y_k^i T_m k + n_k^i \,. \qquad (4)$$

The simulation model for Approaches 2 – 4 will compute the clock phase offsets at discrete times $t_k = kT_m$ (t_k is referred to as time step k). The following are needed:

1) Compute the phase offsets of the free-running node clocks at time step k using Eq. (4). The free-running clock frequency offsets are initialized randomly, from a uniform distribution within a frequency tolerance range, at initialization of the simulation. The clock noise is computed using the model described in the Appendix. Finite clock precision is modeled by truncating the phase offset to a granularity specified as input.

2) At each time step that is a multiple of the positive integer P, i.e., $t_{kP} = kPT_m$, compute an estimate of the frequency offset of each master clock i relative to its slave i+1 using the free-running clock phase offsets. P is specified as an input parameter to the simulation, and is the number of time-stamp intervals (inter-message times) over which the frequency offset is estimated.

3) At each time step kP, calculate the cumulative frequency offset estimate of the grandmaster relative to each slave clock i.

4) Calculate an improved phase offset for each slave clock i+1, using the free-running phase offsets and the most recent cumulative frequency offset estimate relative to the grandmaster.

5) Calculate the clock correction u_k^{i+1} for each slave clock i+1 in terms of the improved phase offsets for the slave clock and its master. Note that the time stamps must contain both the free-running clock times and the improved clock times (based on the computed improved phase offsets).

6) Calculate the cumulative clock correction for each slave clock i+1 relative to the grandmaster, and add this to the improved phase offset calculated in step (5). The result is the unfiltered phase offset for the slave clock.

7) Filter the sequence of unfiltered slave clock offsets with an appropriate digital filter. The result is the filtered phase offset for the slave clock.

If frequency adjustments are not made (Approach 2), then steps (2) – (4) are skipped, and the improved phase offset is set equal to the free-running clock phase offset in step (5). If filtering is not done, then step (7) is skipped and the result of step (6) is the desired clock phase offset.

Step (1) above is performed using Eq. (4) and the phase noise model of the Appendix, along with any specified clock granularity to model finite clock precision. Steps (2) – (7) are now described in detail.

Step 2

The estimate of frequency offset of clock i relative to clock i+1 is equal to the change in time indication of clock i minus the change in time indication of clock i+1, divided by the change in time indication of clock i+1. The changes in time indication are computed over the interval between times $(k-1)PT_m$ and kPT_m. The change in time indication for clock i over this interval is $PT_m + x_{b,kP}^i - x_{b,(k-1)P}^i$. The resulting estimate of frequency offset of clock i relative to clock i+1 is

$$\tilde{y}_{kP}^i = \frac{\left(x_{b,kP}^i - x_{b,(k-1)P}^i\right) - \left(x_{b,kP}^{i+1} - x_{b,(k-1)P}^{i+1}\right)}{PT_m + x_{b,kP}^{i+1} - x_{b,(k-1)P}^{i+1}}$$

$$\tilde{y}_{kP+1}^i = \tilde{y}_{kP+2}^i = \cdots = \tilde{y}_{kP+P-1}^i = \tilde{y}_{kP}^i \qquad , (5)$$

where $\tilde{y}_{kP}^i$ is the frequency offset estimate of clock i relative to clock i+1 at time step kP.

Step 3

The cumulative frequency offset estimate of the grandmaster relative to clock i at time step k, y_k^i, is

$$y_k^i = \sum_{j=1}^{i} \tilde{y}_k^j \,. \qquad (6)$$

Step 4

The improved phase offset estimate for clock i at time step j is calculated under the assumption that the frequency offset of the clock (relative to the grandmaster) is equal to y_j^i since the most recent time step kP when the frequency offset estimate was calculated. With this assumption, the change in corrected time reading of the clock divided by the change in free-running time reading of the clock is equal to one plus the frequency offset estimate, i.e.

$$\frac{(j-kP)T_m + x_j^i - x_{kP}^i}{(j-kP)T_m + x_{b,j}^i - x_{b,kP}^i} = 1 + y_k^i . \quad (7)$$

Then

$$x_j^i = x_{kP}^i + \left(x_{b,j}^i - x_{b,kP}^i\right)\left(1 + y_k^i\right) + (j-kP)T_m y_k^i . \quad (8)$$

Step 5

The clock i+1 correction, u_k^{i+1}, is calculated using Eq. (2), with the slave and master clock times replaced by the respective improved clock phase offsets. Unfortunately, Eq. (8) provides improved phase offset estimates only at the times the slave receives the response time stamps from the master (i.e., the $T_{4,k}$'). An exact calculation of the improved phase offsets at the other time stamp times ($T_{4,k}$', $T_{2,k}$, and $T_{3,k}$) would require implementing the model as a discrete event rather than discrete time (with constant time step) model. Since the latter is computationally much more efficient and therefore desirable, the improved phase offsets at the other times are obtained via interpolation. If the free-running clock frequency offset is the only component of the phase offset (i.e., if there is no phase noise), this approach is valid because the frequency offset gives rise to linear phase variation. If phase noise is present, the approach is only approximate. However, as will be seen in the Appendix, the phase noise model consists of White Phase Modulation (WPM, i.e., white noise) at high frequencies, Flicker Phase Modulation (FPM) at intermediate frequencies, and Flicker Frequency Modulation (FFM) at low frequencies. If the overall level of phase noise in the model is chosen to correspond to the actual noise level sampled at the time stamp rate, then interpolation will give approximately the correct noise level. The result is exact for WPM because it guarantees that the discrete WPM level in the model corresponds to the aliased WPM in the real system when sampled at the time stamp rate. The result is approximate for FPM and FFM because the interpolation corresponds to a filtering of these noise components.

Let r be the fraction of T_m by which the master to slave message is offset from the slave to master message, i.e. $r = x/T_m$, where x is the offset time defined in Figure 5. Also, assume that $D << T_m$ and $D << x$, and take the limit $D \to 0$. Finally, neglect the small probability that a master to slave and slave to master message may overlap in time. Then, replacing the time values by improved phase offset values in Eq. (2), which is permissible because the time values in Eq. (2) all appear as differences, and therefore the fixed (nominal) component of time cancels out

$$\begin{aligned} T'_{4,j} &= x_j^{i+1} \\ T_{3,j} &= x^i(jT_m - D) \cong x_j^i \\ T_{2,j} &= x^i(jT_m - D - x) \cong x^i(jT_m - x) \\ T'_{1,j} &= x^{i+1}(jT_m - 2D - x) \cong x^{i+1}(jT_m - x) \end{aligned} , \quad (9)$$

where $x^i(t)$ represents the improved phase offset for clock i expressed as a function of continuous time.

Finally, the $x^i(jT_m - x)$ are calculated by interpolation

$$\begin{aligned} x^i(jT_m - x) &= x_j^i - \frac{x_j^i - x_{j-1}^i}{T_m} x \\ &= x_j^i(1-r) + x_{j-1}^i r \end{aligned} . \quad (10)$$

As indicated earlier, the time offsets x, and therefore the fractions r, are initialized randomly (from a uniform distribution within [0,1] for r) at initialization of the simulation) and are either held constant during the simulation or change over each time interval T_m by T_m multiplied by the frequency offset between the two clocks.

Inserting Eqs. (9) and (10) into Eq. (2) produces the correction for clock i+1

$$\begin{aligned} u_j^{i+1} &= (1/2)\{x_j^i(1-r) + x_{j-1}^i r \\ &-[x_j^{i+1}(1-r) + x_{j-1}^{i+1}] \\ &-[x_j^{i+1} - x_j^i]\} \\ &= x_j^i\left(1 - \frac{r}{2}\right) - x_j^{i+1}\left(1 - \frac{r}{2}\right) + \frac{r}{2}\left(x_{j-1}^i - x_{j-1}^{i+1}\right) \end{aligned} . \quad (11)$$

Step 6

The cumulative correction for clock i relative to the grandmaster, $u_{k,cum}^i$, is

$$u_{k,cum}^i = \sum_{j=1}^{i} u_k^j . \quad (12)$$

The unfiltered phase offset is equal to $u_{k,cum}^i + x_k^i$.

Step 7

The simulations here use a discrete implementation of a standard second order filter with 20 dB/decade roll-off and gain peaking and 3 dB bandwidth specified on input. The details of the filter and discrete representation are given in Appendix VIII of [15] and in [16]. The standard form for this filter is

$$H(s) = \frac{2\varsigma\omega_n s + \omega_n^2}{s^2 + 2\varsigma\omega_n s + \omega_n^2}$$

ω_n = undamped natural frequency

ς = damping ratio

f_n = 3 dB bandwidth

$= (\omega_n / 2\pi)\left[(2\varsigma^2 + 1) + \sqrt{(2\varsigma^2 + 1)^2 + 1}\right]$

H_p = gain peaking

$= \left[1 - 2\alpha - 2\alpha^2 + 2\alpha\sqrt{2\alpha + \alpha^2}\right]^{-1/2}$,

where $\alpha = 1/(4\varsigma^2)$. (13)

The discrete implementation is based on representing the filter using state variables and writing the filter response as a convolution integral involving the input and state transition matrix. The state transition matrix is evaluated exactly, and the convolution integral over one time step is done analytically using a trapezoidal rule approximation. The filter time step can be taken as a sub-multiple of T_m. See[15] or [16] for details.

5. Simulation cases and results

This section presents the results for six simulation cases. Cases 1 and 2 assume no clock noise and infinite clock precision; Cases 3 – 6 assume clock noise in accordance with the model described in the Appendix and finite clock precision. These are initial cases; future work will consider (among other things) a wider range of parameters and the running of multiple, independent replications for each case to obtain statistical confidence intervals for the results.

Parameters common to all simulation cases are summarized in Table 3. The current HRM for ResE contains a maximum of 7 hops (i.e., 7 Ethernet switches; note that this is not finalized in the ResE Study Group in IEEE 802). The simulation cases here use 10 slave clocks; it was felt desirable to obtain the performance for a synchronization chain that slightly exceeds the ResE HRM. The ±100 ppm frequency tolerance for the free-running slave clocks is standard for Ethernet. The 10 Hz, 0.1 dB filter was felt to be achievable at reasonable cost, and therefore reasonable for initial simulations. The simulation time step was chosen to be a small fraction of the inter-message times (as will be seen shortly, 1 ms for Cases 1 and 2 and 10 ms for Cases 3 – 6) so that phase peaks would be accurately captured for the unfiltered phase variation.

5.1 Cases 1 and 2

Parameters specific to Cases 1 and 2 are given in Table 4. These cases are intended to show the basic performance obtained with (Case 2) and without (Case 1) instantaneous frequency adjustments when no phase noise is present and the clocks have infinite precision. The nominal time between successive time stamps is 1 ms, and the nominal time between successive frequency adjustments in Case 2 is 10 ms.

Figures 6(a) – (b) show unfiltered phase variation for nodes 1 and 10, respectively. Each plot shows the results for Case 1 and Case 2 superimposed. With no frequency adjustments (solid lines), a steady state sawtooth-like phase variation is reached in a time on the order of a few ms. However, the steady-state phase variation is on the order of 70 ns for node 1 and 16 ns for node 10. The phase variation is essentially the cumulative effect of all the frequency offsets of all the clocks over the time-stamp interval (1 ms). For Case 2, there is an initial large transient for the first 10 ms, i.e., the time over which the first frequency measurement is made. However, after the first frequency measurement, a steady-state is reached where the peak-to-peak phase variation is extremely small. While not evident on the scale of the plot, the stead-state peak-to-peak unfiltered phase variation for Case 2 is 0.07 ns.

Figures 6(c) – (d) show the filtered phase variation for Cases 1 – 2. As before, there is an initial transient; however, the time for the transient to decay is now a few filter time constants (1/[2π(10 Hz)] = 16 ms). The steady-state peak-to-peak phase variation is approximately a few tenths of a ns without frequency adjustments and less than 1 ps with frequency adjustments. The extremely good synchronization performance when frequency adjustments are made is due to the fact that, when there is no phase noise and the clocks have infinite precision (zero granularity in phase), the frequency different between master and slave can be measured extremely accurately.

Note that, for a single simulation case, the phase variation does not necessarily increase monotonically as the number of nodes increases; this is because the phase variation depends on the magnitudes and signs of the frequency offsets of the successive clocks. However, if a large number of replications of a simulation case are run and point estimates and confidence intervals for a specific quantile of peak-to-peak phase variation are obtained, it is expected that the quantile point estimate will tend to increase at least over the first few nodes. This will be investigated in more detail in future work.

5.2 Cases 3 and 4

Parameters specific to Cases 3 and 4 are given in Table 4. For these cases, the nominal time between successive time stamps is 10 ms, and the nominal time between successive frequency adjustments in Case 4 is 100 ms (both values are suggested in [4]). Clock phase noise is modeled as described in the Appendix, and the clock phase precision is 1 ns. The time offset

between master to slave and slave to master messages is initialized randomly (between 0 ant T_m) for each node and held constant through the simulation.

Results for filtered and unfiltered phase offset for nodes 1 and 10 are shown in Figures 7(a) – (d). In each figure, the corresponding results for Case 3 (no frequency adjustments) and Case 4 (frequency adjustments) are superimposed. In all cases (i.e., with and without filtering, for both nodes) the steady-state peak-to-peak phase variation for the case with frequency adjustments is much smaller than for the case without frequency adjustments. The peak-to-peak variation of the former is not discernable on the scale of the latter in these figures; therefore, more detailed views of Case 4 are shown in Figures 8(a) – (d). The effect of the noise and granularity is seen here, though the filtered and unfiltered peak-to-peak phase variations are on the order of 1 ns and 2 ns, respectively.

MTIE results for unfiltered and filtered phase variation for selected nodes are shown in Figures 9(b) – (c) (Case 3) and 9(d) – (e) (Case 4), along with the MTIE requirements of Figure 2 (the legend for these plots is given in Figure 9(a)). It is seen that with filtering, MTIE is more than an order of magnitude smaller when frequency adjustments are made than when they are not. For longer observation intervals, the former ranges from 1 – 1.5 ns, while the latter from 10 – 50 ns. For the case with frequency adjustments, the uncompressed digital video MTIE masks are slightly exceeded; without frequency adjustments, these masks and the consumer interface digital audio MTIE mask are exceeded (Figures 9(c) and 9(e)). Without filtering, MTIE ranges from approximately 160 – 600 ns without frequency adjustments and 2 – 4 ns with frequency adjustments. In the latter case, the uncompressed digital video MTIE masks are exceeded; in the former case these and the digital audio masks are exceeded.

As in Cases 1 and 2, the phase variation does not increase monotonically with the number of nodes but, as mentioned in the discussion of those cases, the behavior of MTIE point estimates and confidence intervals as the number of nodes increases must be considered.

5.2 Cases 5 and 6

Parameters specific to Cases 5 and 6 are given in Table 4. The parameters of these cases are the same as in Cases 3 and 4, except that now the time offset between master to slave and slave to master messages is initialized randomly (between 0 ant T_m) for each node and allowed to vary as the simulation proceeds (instead of being held constant). As described in Section 4, this time offset for a master/slave message exchange changes by the frequency offset between the master and slave multiplied by T_m; when the offset reaches either 0 or T_m, it is set to T_m or 0, respectively.

Results for filtered and unfiltered phase offset for nodes 1 and 10 for Case 5 (no frequency adjustments) are shown in Figures 10(a) – (d). When frequency adjustments are not made, the peak-to-peak phase variation can be very large due to phase steps that occur when the master to slave and slave to master messages "walk" past each other, i.e., when the time offset between these messages reaches either 0 or T_m and is adjusted up or down, respectively, by T_m. Comparing Figure 10(a) with 10(c) and Figure 10(b) with 10(d) indicates that the 10 Hz filter greatly attenuates the fast sawtooth due to clock corrections, but cannot reduce the wander due to the steps in master to slave and slave to master message offset time, as the frequency of these steps is very small compared to 10 Hz. In the best case, where the master and slave clock differ by 200 ppm (i.e., one is +100 ppm from nominal and the other -100 ppm from nominal, the offset between the master to slave and slave to master messages changes by $(2 \times 10^{-4})T_m$ over each inter-message time T_m. It therefore requires $T_m/[(2 \times 10^{-4})T_m] = 5000$ inter-message times for a phase step to occur. The phase step frequency is therefore $1/[5000T_m] = 0.02$ Hz, which is very small compared to the 10 Hz filter bandwidth. The peak-to-peak phase variation is on the order of hundreds of ns, with and without filtering.

Results for filtered and unfiltered phase offset for nodes 1 and 10 for Case 6 (with frequency adjustments) are shown in Figures 11(a) – (d). When frequency adjustments are made, the above phase steps do not occur because the error in phase correction due to frequency offset between the master and slave is corrected for. The results for Case 6 are very similar to those for Case 4, where the time offset between master to slave and slave to master messages is fixed throughout the simulation. MTIE results for Cases 5 and 6 are shown in Figures 12(b)-(e) (the legend for the plots is in Figure 12(a)). MTIE for Case 6, for both filtered and unfiltered phase, is very similar to Case 4 (Figures 9(d) and 9(e)). The filtered phase variation MTIE result for Case 6 slightly exceeds the uncompressed digital video mask but meets the other masks; the unfiltered phase variation MTIE result for Case 6 exceeds the uncompressed digital video mask by somewhat more but meets the other masks.

6. Conclusions and future work

In the ideal case of no clock noise and infinite clock precision (i.e., zero granularity), the peak-to-peak phase variation is extremely small. The sawtooth frequency is sufficiently large compared to the 10 Hz filter bandwidth that peak-to-peak phase variation of one or a few tenths of a ns can be achieved even if frequency adjustments are not made; if frequency

adjustments are made, the peak-to-peak phase variation can be less than 1 ps.

For the non-ideal case, with clock noise and finite precision (i.e., noise at the level of the model in the model in the Appendix and 1 ns granularity), the results indicate that filtering is necessary for the uncompressed digital video MTIE masks to be met. The masks are slightly exceeded with the 10 Hz, 0.1 dB gain peaking filter if frequency adjustments are made. In addition, these masks and the digital audio consumer interface mask are exceeded if filtering is done and frequency adjustments are not made. This indicates that the filter bandwidth will likely need to be somewhat narrower. Finally, if the master to slave and slave to master messages are sent at rates traceable to the free-running node clocks rather than at rates traceable to the same clock, the results indicate that peak-to-peak phase variation can be very large (i.e., hundreds of ns) if frequency adjustments are not made.

Additional work is planned to examine the effect of varying the various parameters of the model:

- Filter bandwidth (and possibly gain peaking)
- Time between messages
- Time between frequency adjustments
- Larger clock noise level, which bounds noise in oscillators expected to be used in ResE
- Clock precision (phase granularity)
- Consideration of errors in measurements of times the time stamps are sent and received (modeling cases where the measurements are implemented in the Ethernet PHY, MAC, and above the MAC)

In addition, future simulation cases will have multiple, independent replications run, i.e., at the completion of a run, the state of the random number generator will be saved and used to initialize the next run. This plus the use of a random number generator with a sufficiently large period will ensure that the multiple runs are independent (at least insofar as the pseudo-random samples are independent, i.e., satisfy the necessary statistical tests). The multiple runs will enable confidence intervals for MTIE and other statistics to be obtained.

Finally, additional work is planned to investigate the other approaches in Table 2.

7. References

Note: References [4] – [6], [13], and [14] are available at http://www.ieee802.org/3/re_study/public/index.html.

[1] IEEE Std™ 1588-2002, *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE, November 8, 2002.

[2] Mills, L. David, *A Kernel Model for Precision Timekeeping*, RFC 1589, March, 1994.

[3] Sven Nylund and Oyvind Holmeide, *IEEE 1588 Switch Transparency – No Need for Boundary Clocks!*, 2004 Conference on IEEE 1588, Gaithersburg, MD.

[4] Geoffrey M. Garner, *Description of ResE Video Applications and Requirements*, IEEE 802.3 ResE presentation, Austin, TX, May, 2005.

[5] Geoffrey M. Garner, *Description of ResE Audio Applications and Requirements*, IEEE 802.3 ResE presentation, Austin, TX, May, 2005.

[6] Geoffrey M. Garner, *End-to-End Jitter and Wander Requirements for ResE Applications*, IEEE 802.3 ResE presentation, Austin, TX, May, 2005.

[7] SMPTE 259M-1997, *10-Bit 4:2:2 Component and 4f_{SC} Composite Digital Signals – Serial Digital Interface*, 1997.

[8] ITU-R BT.601-5, *Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide Screen 16:9 Aspect Ratios*, Geneva, 1995.

[9] SMPTE 292M-1998, *Bit-Serial Digital Interface for High-Definition Television Systems*, 1998.

[10] ISO/IEC 13818-1, -2, -3, -9, *Information Technology – Generic Coding of Moving Pictures and Associated Audio Information: Systems (Part 1, 2000), Video (Part 2, 2000), Audio (Part 3, 1996), Extension for Real-Time Interface for Systems Decoders (Part 9, 1996), Geneva.*

[11] IEC-60958-1, -3, -4, *Digital Audio Interface – Part 1: General (2004), - Part 3: Consumer Applications (2003), - Part 4: Professional Applications (2003)*, Geneva.

[12] ITU-T Rec. G.810, *Definitions and Terminology for Synchronization Networks*, Geneva, 1996.

[13] Geoffrey M. Garner, *Summary of Potential Choices for Synchronization Transport in Residential Ethernet*, IEEE 802.3 ResE presentation, San Francisco, CA, July, 2005.

[14] *Residential Ethernet (RE) (a working paper)*, Draft 0.136, maintained by David V. James and based on work by him and other contributors, August 10, 2005.

[15] ITU-T Rec. G.8251, *The Control of Jitter and Wander Within the Optical Transport Network (OTN)*, Geneva, November, 2001, Amendment 1, June, 2002, Corrigendum 1, June, 2002.

[16] Geoffrey M. Garner, *Jitter Analysis for Asynchronous Mapping of a Client Signal into an OcH,* Lucent Technologies Contribution to ITU-T Q 11/15 Interim Meeting, Ottawa, ON, July, 2000.

[17] *Phase Noise*, Vectron International, available at http://www.vectron.com

[18] *Jitter and Signal Noise in Frequency Sources*, Raltron Application Note, available at http://www.raltron.com.

[19] Dan H. Wolaver, *Phase-Locked Loop Circuit Design*, Prentice-Hall, 1991, Chapter 6.

[20] David W. Allan, Marc A. Weiss, and James L. Jespersen, *A Frequency Domain View of Time Domain Characterization of Clocks andTime and Frequency Distribution Systems*, Forty-Fifth Annual Symposium of Frequency Control, Los Angeles, CA, May 29 – 31, 1991, pp. 667 – 678.

[21] Stefano Bregni, *Synchronization of Digital Telecommunications Networks*, Wiley, 2002.

[22] J.A. Barnes and Stephen Jarvis, *Efficient Numerical and Analog Modeling of Flicker Noise Processes*, National Bureau of Standards, NBS Technical Note 604, June, 1971.

[23] James A. Barnes and Charles A. Greenhall, *Large Sample Simulation of Flicker Noise*, 19th Annual Precision

Time and Time Interval (PTTI) Applications Planning Meeting, December, 1987.
[24] Giovanni Corsini and Robert Saletti, *A 1/f^γ Power Spectrum Noise Sequence Generator*, IEEE Transactions on Instrumentation and Measurement, Vol. 37, No. 4, December, 1988, pp. 615 – 619.

Appendix. Clock noise model

Clock phase noise may be modeled as a sum of random processes with power spectral density (PSD) of the form

$$S(f) = \frac{A}{f^{\alpha}}, \qquad \text{(A-1)}$$

where A is a constant and α is typically a small integer. In practice, the PSD has 3 terms [17] – [19]

- $\alpha = 0$, White Phase Modulation (WPM), i.e., white noise
- $\alpha = 1$, Flicker Phase Modulation (FPM)
- $\alpha = 3$ Flicker Frequency Modulation (FFM)

The PSD of the phase noise can be written

$$S_x(f) = \frac{A}{f^3} + \frac{B}{f} + C, \qquad \text{(A-2)}$$

where $S_x(f)$ has units ns^2/Hz. (Note that if a term with $\alpha = 2$ were present, it would be referred to as White Frequency Modulation (WFM), i.e., a random walk in phase; however, a WFM component is not included in the model here.)

Often an alternate form of the PSD is used

$$S_{\phi}(f) = (2\pi\nu_0)^2 S_x(f), \qquad \text{(A-3)}$$

where $S_\phi(f)$ has units rad^2/Hz. An example specification, taken from Figure 12 of [18], is shown in Figure A-1. Note that the data in [18] is given for a two-sided PSD in units of dBc/Hz (decibels relative to the carrier); the data is converted to the above one-sided PSD expressed in rad^2/Hz using the conversion

$$S(f)\big|_{2-\text{sided}}(\text{dBc/Hz}) = \frac{1}{2} S(f)\big|_{1-\text{sided}}(\text{rad}^2/\text{Hz}), \qquad \text{(A-4)}$$

Note that the data in [18] does not extend to frequencies above 10 kHz; it is conservatively assumed here that the PSD does not decrease further for higher frequencies, and instead remains flat at the 10 kHz value (i.e., the WPM region is assumed to begin at 10 kHz). Finally, the data in [18] has the piecewise linear (i.e., linear on a log scale) form given by the dotted curve; this is conservatively approximated by the solid curve, which is a fit if Eq. (A-2). The specifications provided for specific devices of [17] and [17] are below the example PSD of Figure 12 of [18] (i.e., the data that gives rise to Figure A-1 here), at least for devices for which phase noise specifications are provided.

Another measure for clock phase noise is Time Variance (TVAR), along with its square root, Time Deviation (TDEV). TVAR and TDEV are more convenient than PSD because they are time domain parameters whose estimates are easily calculated from measured or simulated phase data. TVAR is defined as 1/6 times the expectation of the square of the second difference of the phase error averaged over an interval [12]

$$\text{TVAR}(\tau) = \frac{1}{6} E\left[\left(\Delta^2 \bar{x}\right)^2\right], \qquad \text{(A-5)}$$

where $E[\cdot]$ denotes expectation, $\bar{x}$ denotes average over the integration time τ, and Δ^2 denotes second difference. TVAR may be estimated from measured or simulated data using [12]

$$\text{TVAR}(n\tau_0) = \frac{1}{6n^2(N-3n+1)} \cdot \sum_{j=1}^{N-3n+1}\left[\sum_{i=j}^{n+j-1}\left(x_{i+2n} - 2x_{i+n} + x_i\right)\right]^2, \quad n = 1, 2, ..., \text{integer part}(N/3). \qquad \text{(A-6)}$$

where τ_0 is the sampling interval and $\tau = n\tau_0$. TVAR is equal to $\tau^2/3$ multiplied by the Modified Allan Variance [12].

For power-law noises with PSD proportional to $1/f^{\alpha}$, TVAR is proportional to τ^{β}, with $\beta = \alpha - 1$ [12], [20], [21]. In addition, the magnitude of TVAR may be related to the magnitude of PSD for power-law noises [20], [21]

FFM

$$S_x(f) = \frac{A}{f^3} \qquad \text{TVAR}(\tau) = \frac{(2\pi)^2 9 \ln 2}{20} A\tau^2 \qquad \text{(A-7)}$$

FPM (result is from [20]; a more exact expression is given in [21])

$$S_x(f) = \frac{B}{f} \quad \text{TVAR}(\tau) = \frac{3.37}{3} B \quad \text{(A-8)}$$

WPM

$$S_x(f) = C \quad \text{TVAR}(\tau) = \frac{\tau_0 f_h}{\tau} C \quad \text{(A-9)}$$

$$f_h = \text{noise bandwidth}$$

The above expressions are used to obtain TDEV(τ) that is equivalent to the PSD of Figure A-1 (and represented by an expression of the form of Eq. (A-2). Each noise component is simulated separately and added to obtain the total phase noise. To verify that the noise model matches the PSD, TDEV for the simulated noise samples is evaluated and compared with the TDEV mask equivalent to the PSD. We now briefly explain how each noise component is simulated.

WPM is simulated as a sequence of independent, identically distributed random samples. The noise distribution is taken as Gaussian with zero mean. The noise variance and sampling time determine the TDEV level. The variance is chosen such that, with the given sampling time, the computed TDEV from a sample history matches the value obtained from Eq. (A-9) above. The noise bandwidth is assumed to be equal to the line rate, i.e., 100 MHz.

FPM is simulated by passing a sequence of independent, identically distributed, Gaussian random samples through a Barnes/Jarvis filter [22] – [24]. If white noise is input to a filter with frequency response $H(f) = f^{-1/2}$, the output is a random process with PSD proportional to $1/f$, i.e., FPM. The Barnes/Jarvis filter approximates an $f^{-1/2}$ frequency response using a bank of lead/lag filters. The actual frequency response of this filter resembles a "staircase." The spacing of the poles and zeros are chosen such that the average slope is –10 dB/decade. The accuracy obtained depends on the number of poles and zeros per decade of frequency; see [24] for details. The variance of the Gaussian random samples input to the filter determines the TDEV level; the variance is chosen such that the computed TDEV from the resulting filter output matches the value obtained from Eq. (A-8) above.

FFM is simulated by passing a sequence of independent, identically distributed, Gaussian random samples (with zero mean) through a Barnes/Jarvis filter followed by an integrator (accumulator). As with FPM, the variance of the Gaussian distribution determines the TDEV level. The variance is chosen such that the computed TDEV from the resulting filter output matches the value obtained from Eq. (A-7) above.

TDEV for a sample history of simulated data (using the above modeling procedures) and the TDEV mask equivalent to the PSD of Figure A-1 are shown in Figure A-2. The time step for the simulation is 0.01 ms (the same as in the simulation cases described in Section 5).

Table 1. End-to-end jitter and wander requirements for A/V applications

Requirement	Uncompressed SDTV	Uncompressed HDTV	MPEG-2, with network transport	MPEG-2, no network transport	Digital audio, consumer interface	Digital audio, professional interface
Wide-band jitter (UIpp)	0.2	1.0	50 µs peak-to-peak phase variation requirement (no measurement filter specified)	1000 ns peak-to-peak phase variation requirement (no measurement filter specified)	0.25	0.25
Wide-band jitter meas filt (Hz)	10	10			200	8000
High-band jitter (UIpp)	0.2	0.2			0.2	No requirement
High-band jitter meas filt (kHz)	1	100			400 (approx)	No requirement
Frequency offset (ppm)	±2.79365 (NTSC) ±0.225549 (PAL)	±10	±30	±30	±50 (Level 1) ±1000 (Level 2)	±1 (Grade 1) ±10 (Grade 2)
Frequency drift rate (ppm/s)	0.027937 (NTSC) 0.0225549 (PAL)	No requirement	0.000278	0.000278	No requirement	No requirement

Table 2. Summary of approaches for using slave clock corrections in performing synchronization

	Approach
1	Use one-way time stamp scheme with less frequent two-way exchange; obtain delay from two-way exchange and assume delay is fixed until next two-way exchange
2	Instantaneous phase adjustments at intermediate nodes
3	Instantaneous phase and frequency adjustments at intermediate nodes, with frequency adjustments possibly less frequent [4]
4	Filtered phase adjustments at intermediate nodes, using digital filter running at local clock rate (with or without instantaneous frequency adjustments)
5	Full phase-locked loops (PLLs) at intermediate nodes
6	Use of transparent clocks [3] a) end-to-end versus peer-to-peer b) whether or not to adjust rate of local oscillator in transparent clock and, if so, whether to do filtering
7	Time stamp reflecting current time versus delay by some number of frames
8	Time stamp reflects local free-running clock time versus latest corrected time based on most recent time stamps and possible filtering

Table 3. Parameters common to all simulation cases

Common Parameters
10 hops (grandmaster followed by chain of 10 slave clocks)
Slave clock frequency tolerance = ± 100 ppm
Filter bandwidth = 10 Hz
Filter gain peaking = 0.1 dB
Simulation time step = 0.01 ms (used small time step to ensure phase peaks were captured)

Table 4. Parameters specific to each simulation case

Case	Include phase noise	Clock precision (ns)	Instant-aneous freq-uency adjust-ments	Time offset between M/S and S/M messages	Inter-message time (ms)	Time between frequency updates (ms)
1	**No**	**0**	**No**	**Fixed**	**1**	**–**
2	**No**	**0**	**Yes**	**Fixed**	**1**	**10**
3	**Yes**	**1**	**No**	**Fixed**	**10**	**–**
4	**Yes**	**1**	**Yes**	**Fixed**	**10**	**100**
5	**Yes**	**1**	**No**	**Vary-ing**	**10**	**–**
6	**Yes**	**1**	**Yes**	**Vary-ing**	**10**	**100**

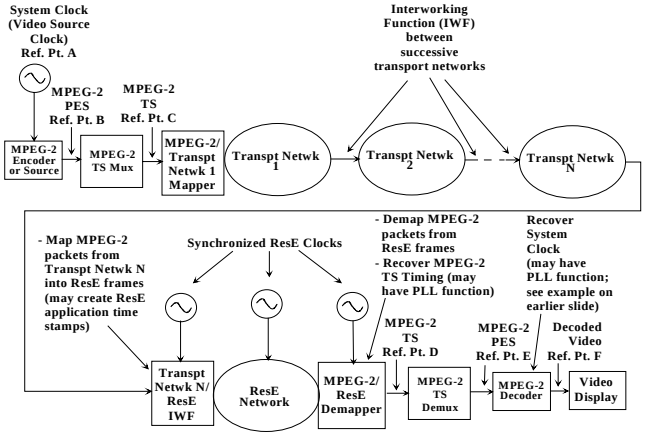


Fig. 1. Reference model for transport of MPEG-2 video over service provider networks and Residential Ethernet

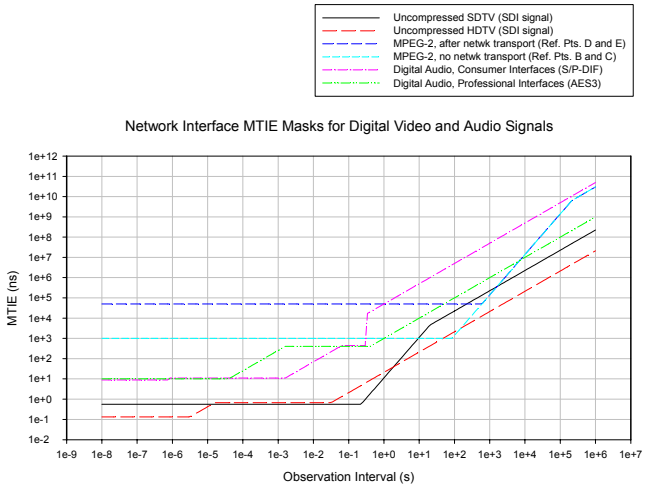


Fig. 2. Jitter and wander MTIE masks for digital video and audio applications (Reference Points are defined in Fig. 1)

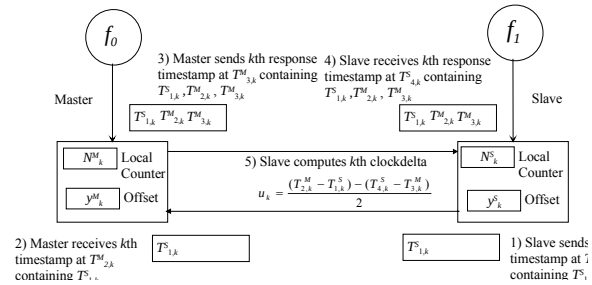


Fig. 3. Illustration of time stamp exchange between master and slave clock and computation of correction for slave

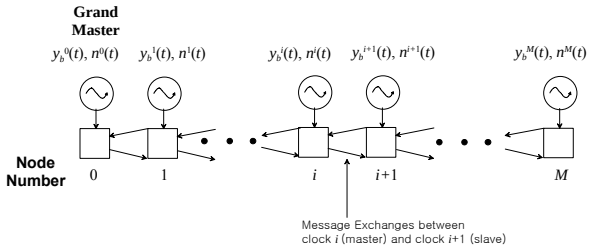


Fig. 4. Model for synchronization of a chain of slave clocks, using Approaches 2 – 4 of Table 2

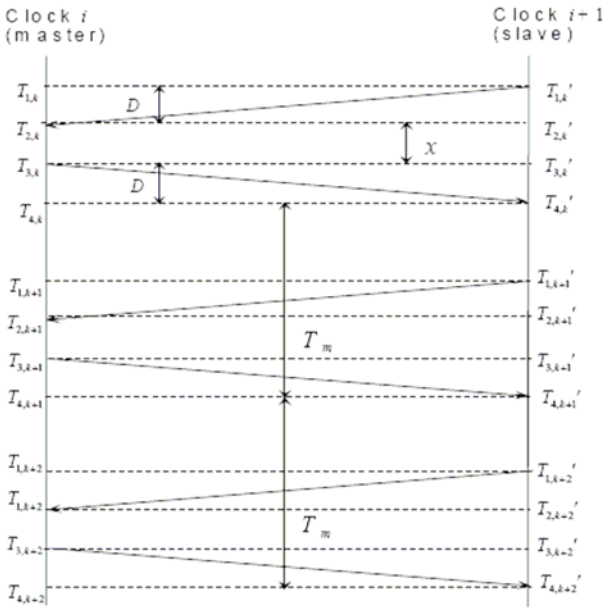


Fig. 5. Details of 3 successive message exchanges between a slave and master clock

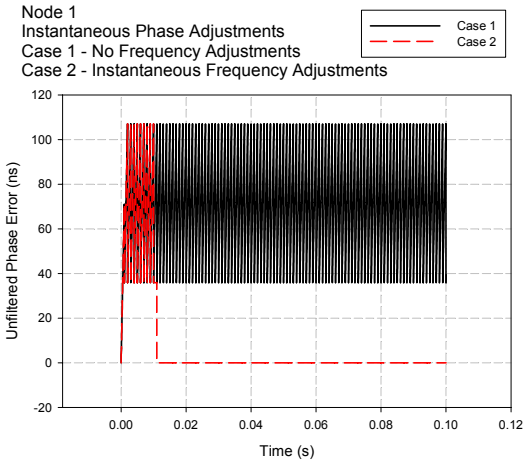


Fig. 6(a). Unfiltered phase offset for Cases 1 and 2, node 1

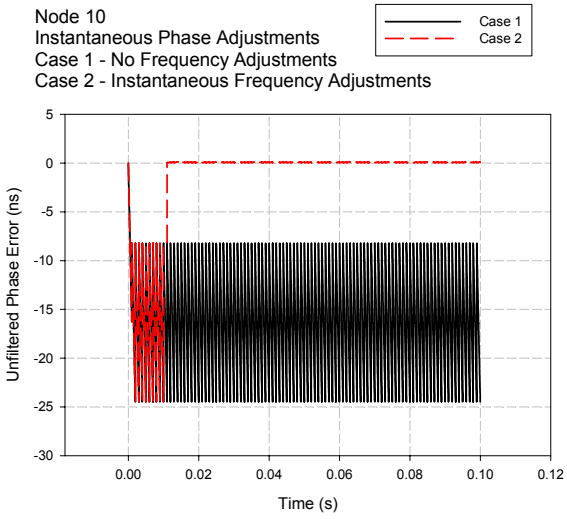


Fig. 6(b). Unfiltered phase offset for Cases 1 and 2, node 10

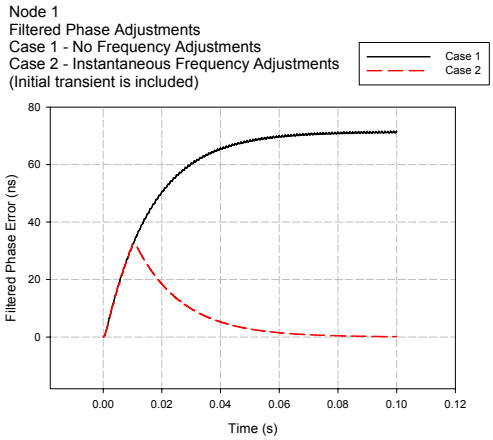


Fig. 6(c). Filtered phase offset for Cases 1 and 2, node 1

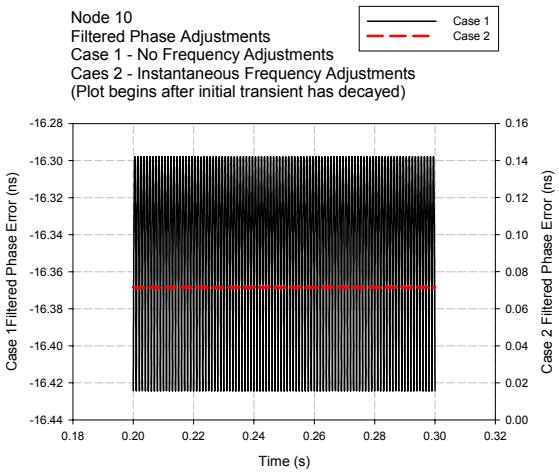


Fig. 6(d). Filtered phase offset for Cases 1 and 2, node 10

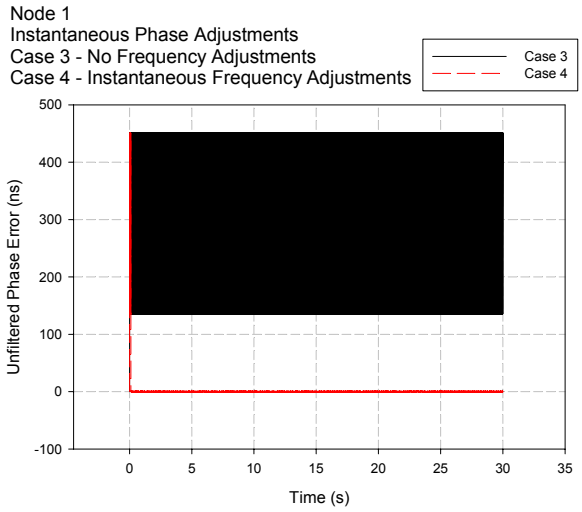


Fig. 7(a). Unfiltered phase offset for Cases 3 and 4, node 1

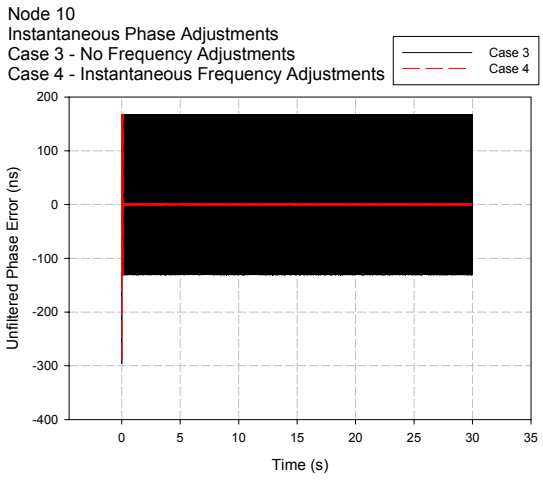


Fig. 7(b). Unfiltered phase offset for Cases 3 and 4, node 10

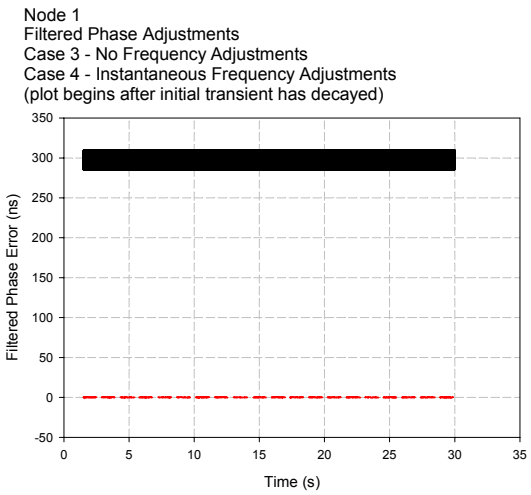


Fig. 7(c). Filtered phase offset for Cases 3 and 4, node 1

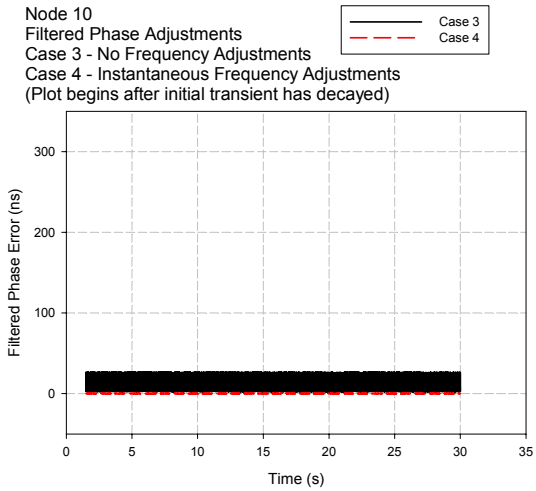


Fig. 7(d). Filtered phase offset for Cases 3 and 4, node 10

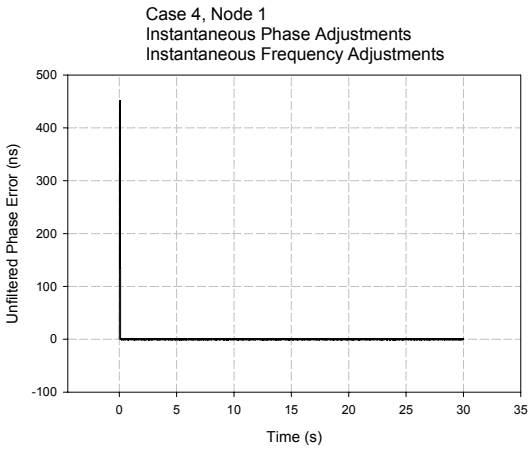


Fig. 8(a). Detailed view of unfiltered phase offset for Case 4, node 1

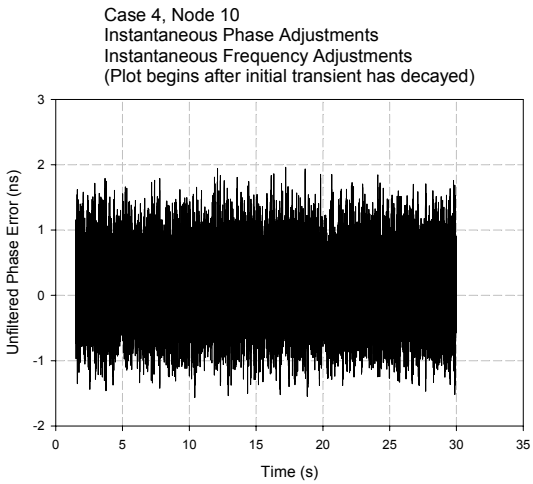


Fig. 8(b). Detailed view of unfiltered phase offset for Case 4, node 10

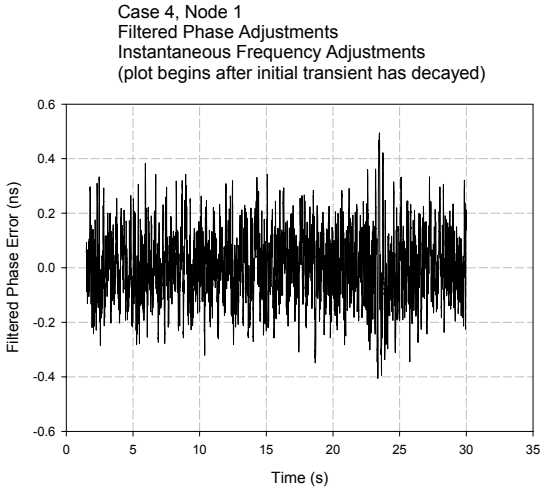


Fig. 8(c). Detailed view of filtered phase offset for Case 4, node 1

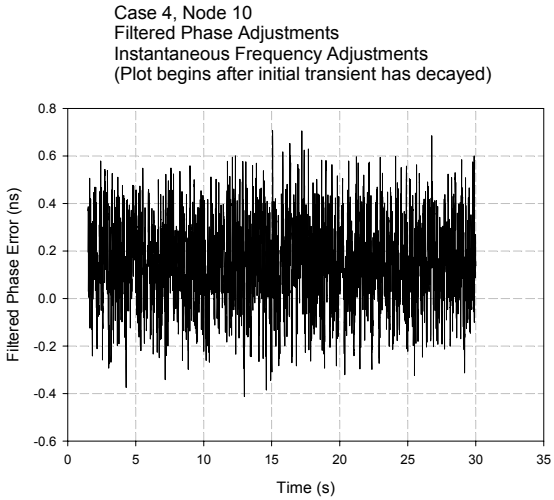


Fig. 8(d). Detailed view of filtered phase offset for Case 4, node 10

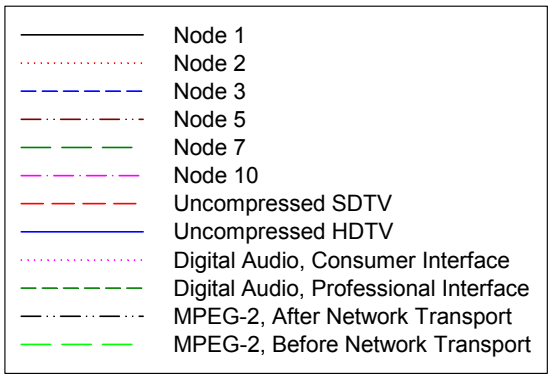


Fig. 9(a). Legend for Figs. 9(b) – (e)

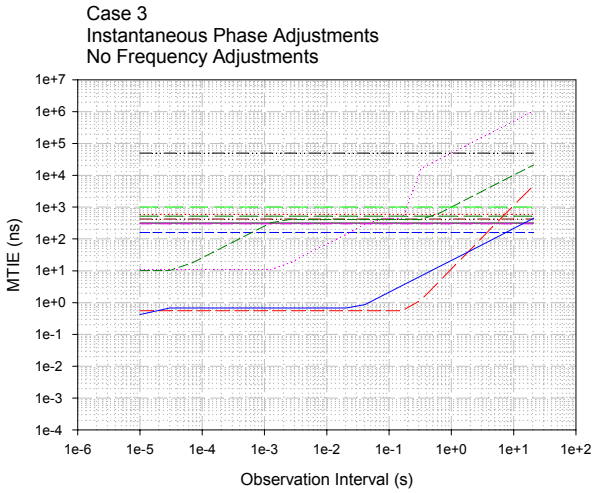


Fig. 9(b). MTIE, Case 3, unfiltered phase variation

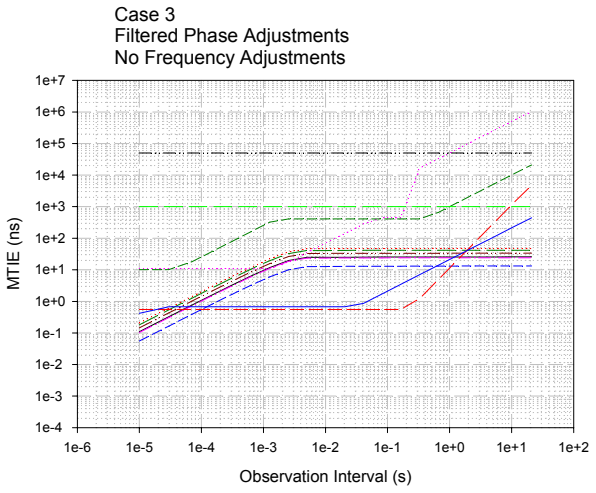


Fig. 9(c). MTIE, Case 3, filtered phase variation

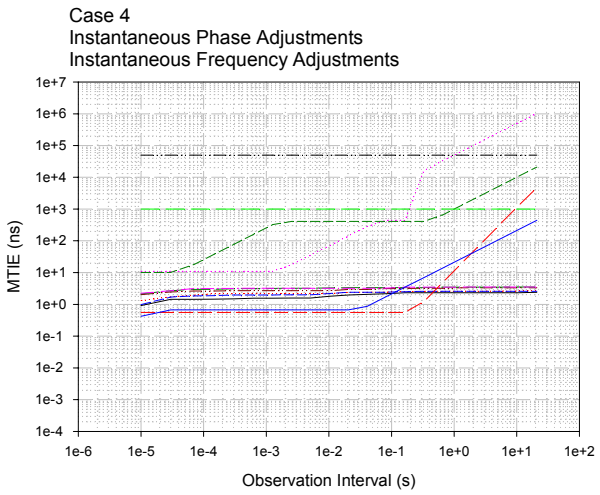


Fig. 9(d). MTIE, Case 4, unfiltered phase variation

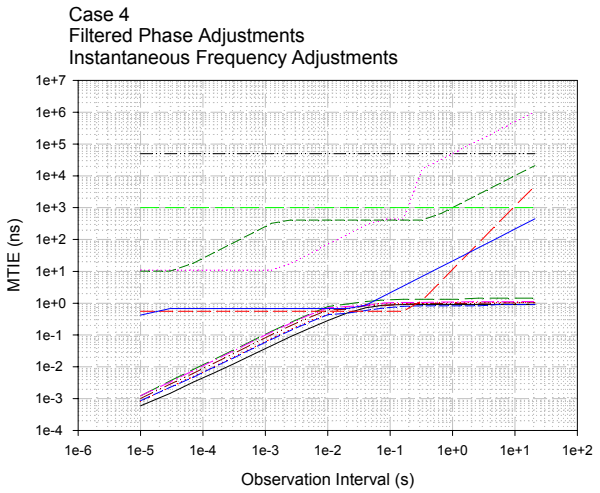


Fig. 9(e). MTIE, Case 4, filtered phase variation

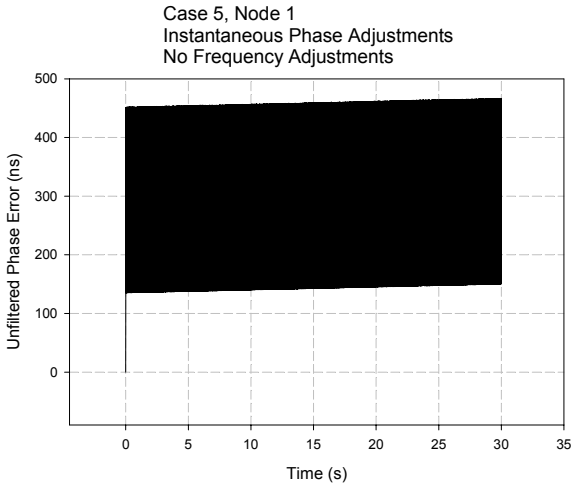


Fig. 10(a). Unfiltered phase offset for Case 5, node 1

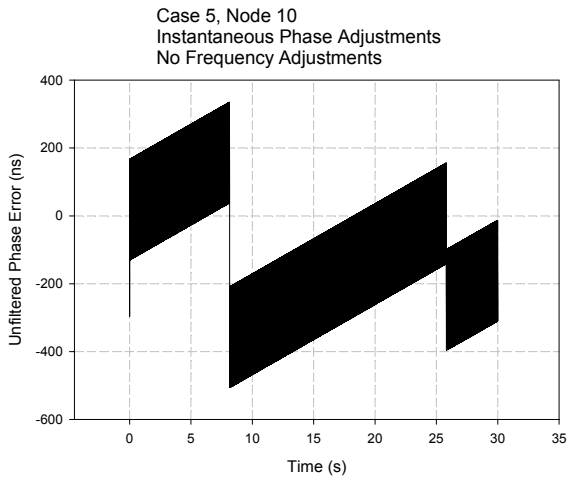


Fig. 10(b). Unfiltered phase offset for Case 5, node 10

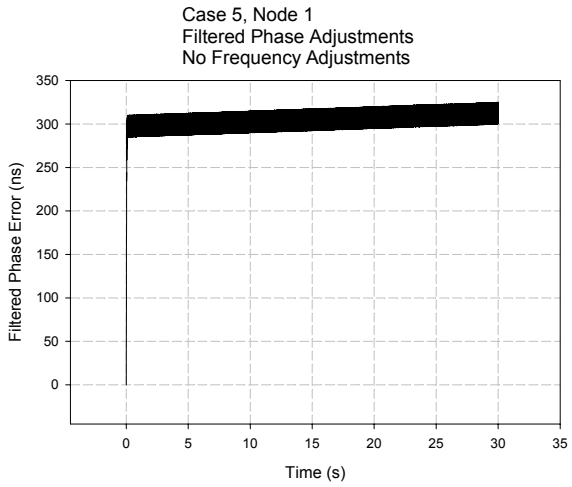


Fig. 10(c). Filtered phase offset for Case 5, node 1

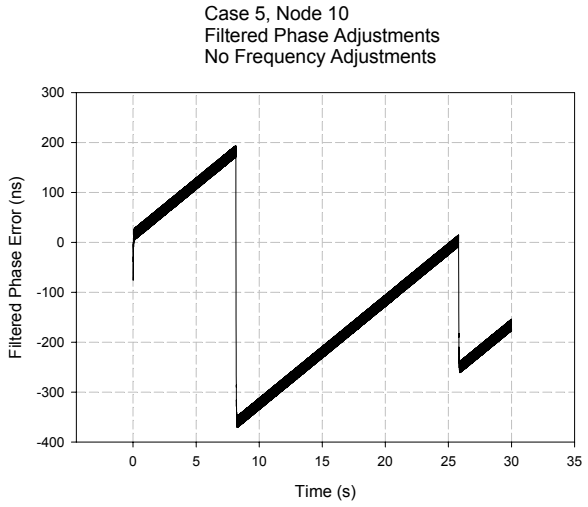


Fig. 10(d). Filtered phase offset for Case 5, node 1

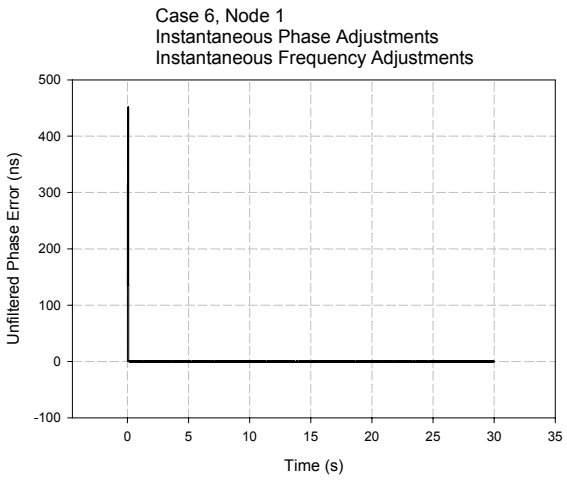


Fig. 11(a). Unfiltered phase offset for Case 6, node 1

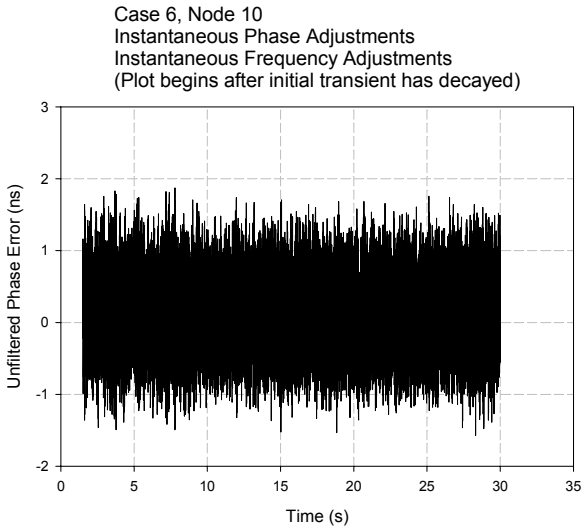


Fig. 11(b). Unfiltered phase offset for Case 6, node 10

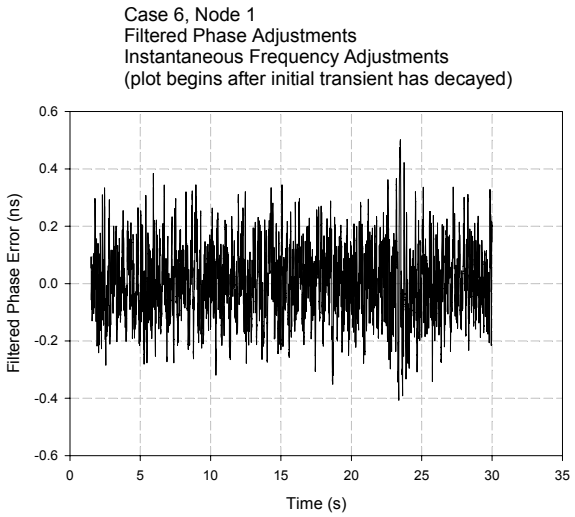


Fig. 11(c). Filtered phase offset for Case 6, node 1

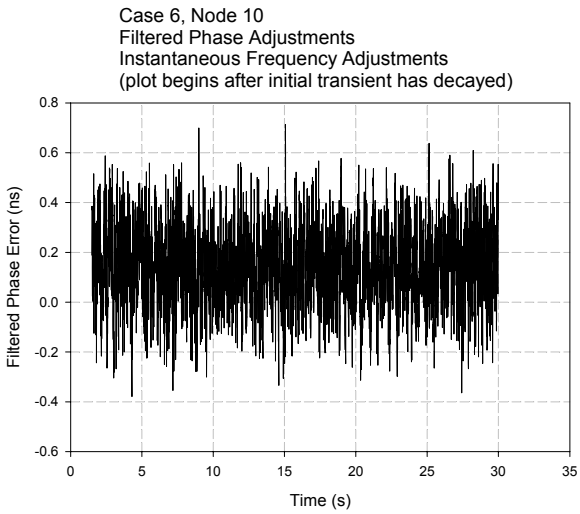


Fig. 11(d). Filtered phase offset for Case 6, node 10

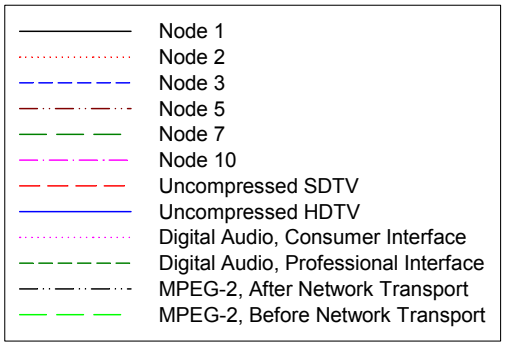


Fig. 12(a). Legend for Figs. 12(b) – (e)

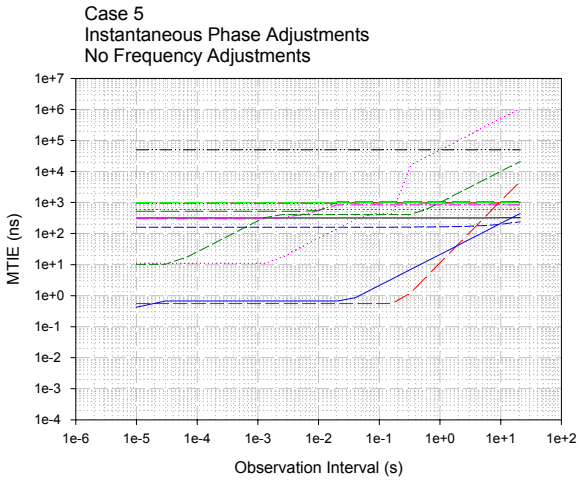


Fig. 12(b). MTIE, Case 5, unfiltered phase variation

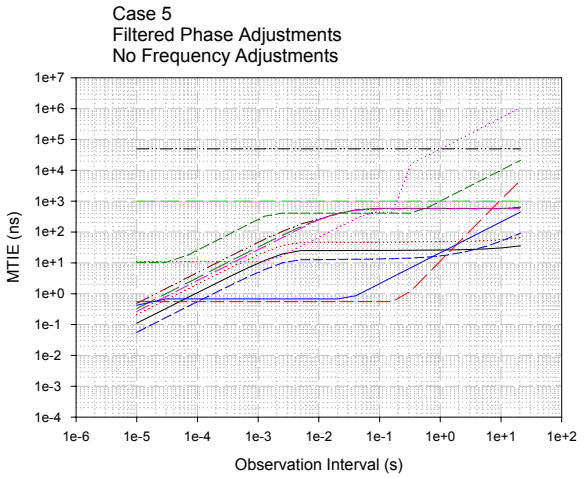


Fig. 12(c). MTIE, Case 5, filtered phase variation

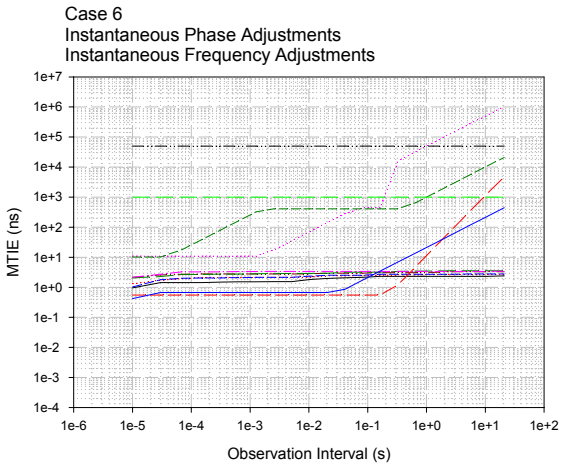


Fig. 12(d). MTIE, Case 6, unfiltered phase variation

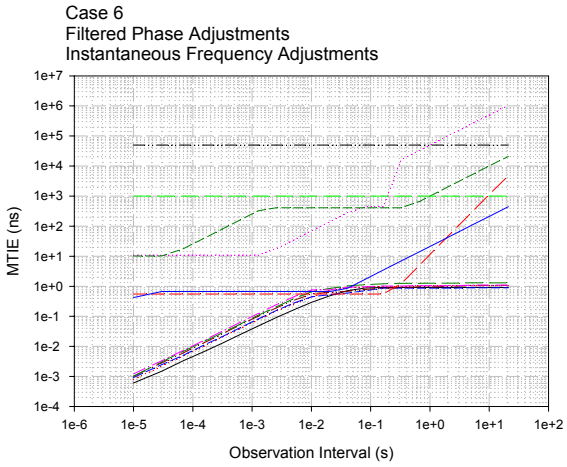


Fig. 12(e). MTIE, Case 6, filtered phase variation

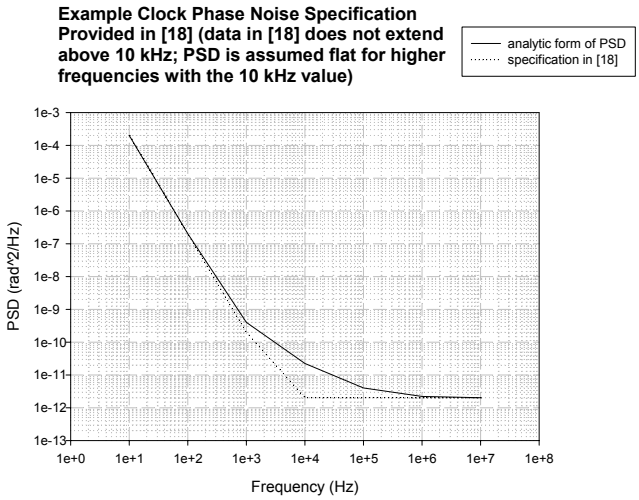


Fig. A-1. Example clock phase noise specification, taken from [18]. Note that data in [18] is given in dBc/Hz, and has been converted to rad^2/Hz.

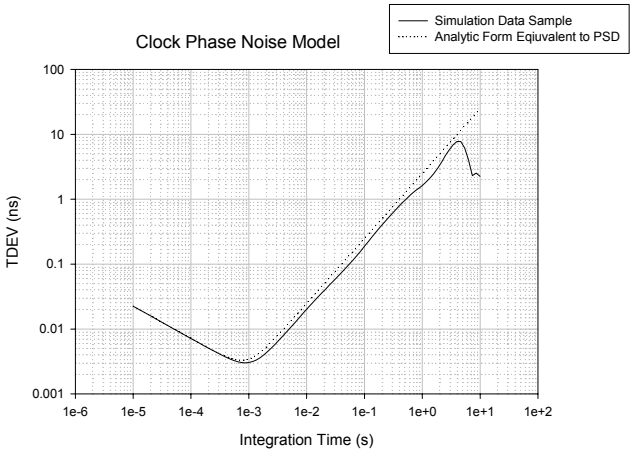


Fig. A-2. TDEV equivalent to PSD of Figure A-1, and TDEV computed from equivalent simulation model data.

Leading the Next

Analysis of Clock Synchronization Approaches for Residential Ethernet

Geoffrey M. Garner
Kees den Hollander
SAIT / SAMSUNG Electronics
gmgarner@comcast.net
denhollander.c.j@samsung.com

2005 Conference on IEEE 1588
October 10 – 12, 2005
Winterthur, Switzerland

SAMSUNG

Outline

Leading the Next

- Introduction
- Application reference models
- End-to-end requirements
- Synchronization approaches for ResE
- Synchronization model
- Simulation cases and results
- Conclusions
- Future work
- References
- Appendix I – Clock noise model
- Appendix II – Definition of MTIE

Introduction

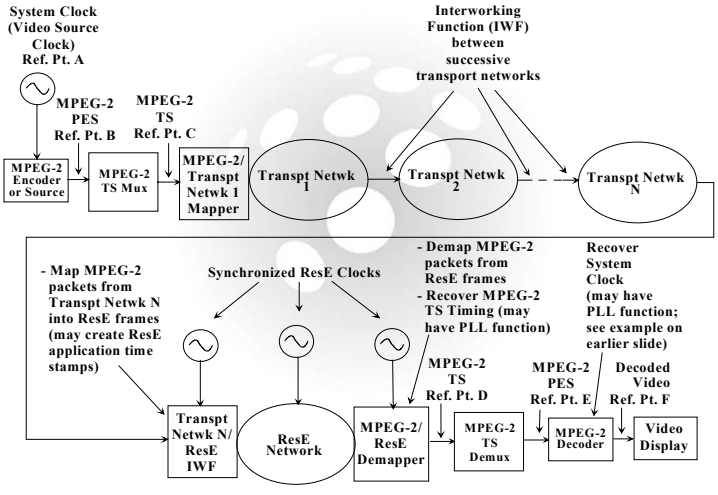
Leading the Next

- Residential Ethernet (ResE) is a new standardization activity in IEEE 802 that is considering extensions to Ethernet to allow the transport of time-sensitive traffic (e.g., high quality audio and video (A/V))
- A/V applications have tight jitter and wander requirements that must be met end-to-end
- To meet these requirements, synchronization is required at ResE ingress and egress points
- This analysis investigates if and how synchronization approaches based on IEEE 1588 can meet the ResE requirements

Application Reference Models

Leading the Next

Example Reference Model for Transport of MPEG-2 Video over Service Provider Networks and Residential Ethernet [2]



End-to-End Requirements

Leading the Next

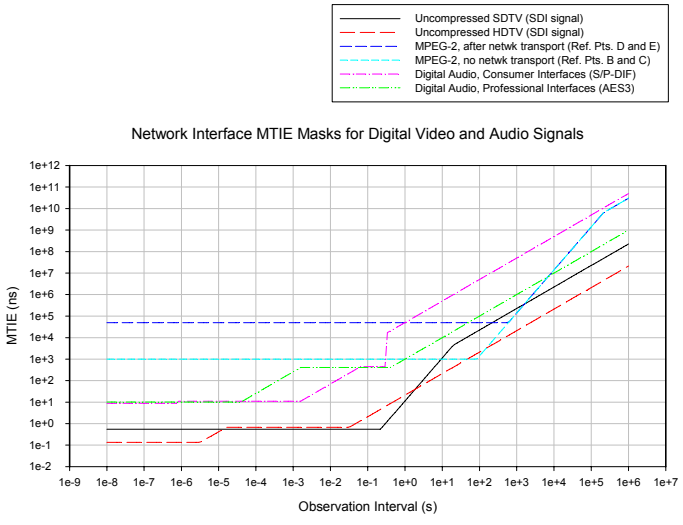
Summary of End-to-End Application Jitter and Wander Requirements (see[2] and references given there)

Requirement	Uncompressed SDTV	Uncompressed HDTV	MPEG-2, with network transport	MPEG-2, no network transport	Digital audio, consumer interface	Digital audio, professional interface
Wide-band jitter (UIpp)	0.2	1.0	50 μs peak-to-peak phase variation requirement (no measurement filter specified)	1000 ns peak-to-peak phase variation requirement (no measurement filter specified)	0.25	0.25
Wide-band jitter meas filt (Hz)	10	10			200	8000
High-band jitter (UIpp)	0.2	0.2			0.2	No requirement
High-band jitter meas filt (kHz)	1	100			400 (approx)	No requirement
Frequency offset (ppm)	±2.79365 (NTSC) ±0.225549 (PAL)	±10	±30	±30	±50 (Level 1) ±1000 (Level 2)	±1 (Grade 1) ±10 (Grade 2)
Frequency drift rate (ppm/s)	0.027937 (NTSC) 0.0225549 (PAL)	No requirement	0.000278	0.000278	No requirement	No requirement

End-to-End Requirements

Leading the Next

End-to-End Application Jitter and Wander Requirements Expressed as MTIE Masks [2] (see Appendix II for MTIE definition)

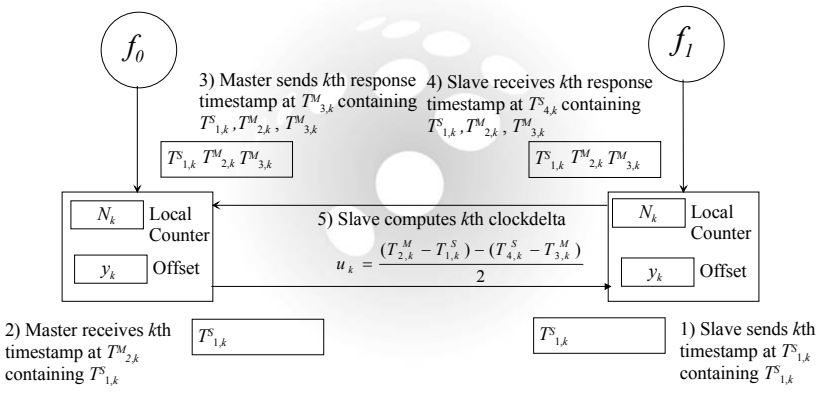


Synchronization Approaches

Leading the Next

Basic 2-Way Time Stamp Approach used in IEEE 1588

- ResE will use this basic approach; however, a number of variations are possible
- Generally assumed a filtering function will be present at the endpoint
  - May be present at intermediate nodes (i.e., in some variations)



6) Slave computes current offset y_k in terms of current and possibly past clockdelta's u_k

Synchronization Approaches

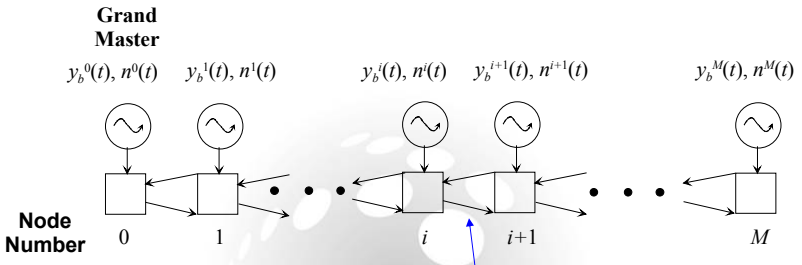
Leading the Next

Variations/Choices

1) Use one-way time stamp scheme with less frequent two-way exchange; obtain delay from two-way exchange and assume delay is fixed until next two-way exchange
2) **Instantaneous phase adjustments at intermediate nodes**
3) **Instantaneous phase and frequency adjustments at intermediate nodes (with instantaneous frequency adjustments possibly less frequent)**
   - **Described in [4]**
4) **Filtered phase adjustments at intermediate nodes, using digital filter running at local clock rate (with or without instantaneous frequency adjustments)**
5) Full phase-locked loops (PLLs) at intermediate nodes (i.e., filtered phase and frequency adjustments)
6) Use of transparent clock nodes
   a) End-to-end versus peer-to-peer
   b) Whether or not to adjust rate of local oscillator in transparent clock and, if so, whether to do filtering
7) Time stamp reflects current time versus delay by some number of frames
8) Time stamp reflects local free-running clock time versus latest corrected time based on most recent time stamps and possible filtering)

Synchronization Model

Leading the Next



$x_b^i(t)$ = phase offset of clock i relative to UTC (ns)

$y_b^i(t)$ = frequency offset (pure fraction) of clock i relative to UTC

$n^i(t)$ = phase noise of clock i (ns)

$x_b^i(i) = \int_0^t y_b^i(t)\,dt + n^i(t) = y_b^i t + n^i(t)$

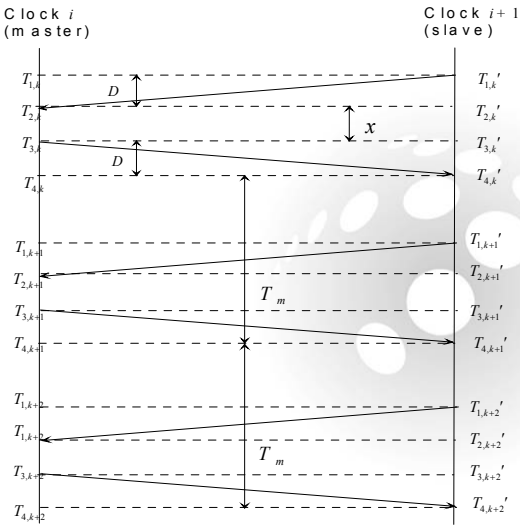
- Assumes the frequency offset is constant over time
- Assumes the phase offsets are zero at $t = 0$
- We are interested in timing relative to the GM;
  therefore, can set $y_b^0(t) = n^0(t) = 0$

Message Exchanges between clock i (master) and clock i+1 (slave)

-Note that messages from master to slave and slave to master do not necessarily occur at the same times
-Note that messages from master to slave and slave to master may not occur at the same rates

Synchronization Model

Leading the Next



T_m = time between successive messages from master to slave, measured relative to UTC

D = propation delay between master and slave

x = time offset between master and slave (will be initialized randomly between 0 and T_m and either kept constant or allowed to change by frequency offset between master and slave multiplied by T_m

- Assume $D << T_m$, and therefore probablity that messages from master to slave and slave to master overlap in time is negligible
- Unprimed quantities are relative to master clock
- Primed quantities are relative to slave clock

Then, can express the phase offset in discrete time (k = time index; UTC time at step $k = kT_m$)

$x_{b,k}^i = y_s^i T_m k + n_k^i$

Synchronization Model

Leading the Next

- Outline of model derivation
  - For variations (3) and (4), express frequency offset estimate of slave relative to master over *P* time steps in terms of the $x_{b,k}{}^{i}$ and $x_{b,k}{}^{i+1}$ (tilde denotes relative frequency offset between current and previous node)
    - Compare time differences in free-running master and slave clocks over PT_m

$$\tilde{y}_{kP}^{i} = \frac{\left(PT_m + x_{b,kP}^{i} - x_{b,(k-1)P}^{i}\right) - \left(PT_m + x_{b,kP}^{i+1} - x_{b,(k-1)P}^{i+1}\right)}{PT_m + x_{b,kP}^{i+1} - x_{b,(k-1)P}^{i+1}}$$

$$\tilde{y}_{kP+1}^{i} = \tilde{y}_{kP+2}^{i} = \cdots = \tilde{y}_{kP+P-1}^{i} = \tilde{y}_{kP}^{i}$$

  - For variations (3) and (4), calculate cumulative frequency offset of current node relative to GM $y_k^i = \sum_{j=1}^{i} \tilde{y}_k^j$
  - For variation (3) and (4), express corrected phase error estimate $x_j{}^{i}$ in terms cumulative frequency offset estimate and free-running clock phase error $x_{b,j}{}^{i}$
    - Choose phase error estimate at all time steps between frequency updates to be consistent with current frequency offset estimate

$$\frac{(j-k)PT_m + x_j^i - x_{kP}^i}{(j-k)PT_m + x_{b,j}^i - x_{b,kP}^i} = 1 + y_k^i$$

$$x_j^i = x_{kP}^i + \left(x_{b,j}^i - x_{b,kP}^i\right)\left(1 + y_k^i\right) + (j-kP)T_m y_k^i$$

Synchronization Model

Leading the Next

- For variations (2) – (4), calculate clock delta in terms of either corrected phase error estimates (for cases where frequency adjustments are made) or free-running clock phase errors
  - Apply result for clock delta in step (5) on slide 7
  - Need phase error values at intermediate times $T_{1,k}', T_{2,k}, T_{3,k}$
  - Obtain these by interpolation; result depends on x and D
    - Take limit $D \to 0$
    - Note: assumption is being made that we can interpolate on the noise
      - –Reasonable as long as the desired noise level is chosen for sampling rate T_m
  - See paper for details
- Calculate cumulative clock delta for all nodes up to the current one (GM clock delta is zero)
- Add cumulative clock delta to corrected or free-running clock phase error to obtain unfiltered phase estimate
- Filter the unfiltered phase estimate with a digital filter that runs at the local clock rate

Synchronization Model

Leading the Next

- Since the filter is linear, the result is the same for the case where each clock delta is filtered at each respective intermediate node versus filtering the cumulative clock delta
  - If synchronization is needed at each node, the work is the same in either case
- Filter model is a digital implementation of standard 2nd order, linear filter with 20 dB/decade roll-off

$$H(s) = \frac{2\varsigma\omega_n s + \omega_n^2}{s^2 + 2\varsigma\omega_n s + \omega_n^2}$$

ω_n = undamped natural frequency

ς = damping ratio

$$f_n = 3\text{ dB bandwidth} = (\omega_n / 2\pi)\left[(2\varsigma^2 + 1) + \sqrt{(2\varsigma^2 + 1)^2 + 1}\right]$$

$$H_p = \text{gain peaking} = \left[1 - 2\alpha - 2\alpha^2 + 2\alpha\sqrt{2\alpha + \alpha^2}\right]^{-1/2}$$

where $\alpha = 1/(4\varsigma^2)$

- The digital implementation is obtained by expressing the filter in state variable form (See [6] and [7] for details)
  - State vector at current time step is written as convolution integral of input vector and impulse response matrix
  - Impulse response matrix is calculated exactly and integral is evaluated using trapezoidal approximation for input
  - Output is written in terms of states

Synchronization Model

Leading the Next

- Additional aspects of model
  - Clock noise model is described in appendix
  - Simulation time step is a sub-multiple of the inter-message time T_m (cannot exceed T_m)
  - Time between frequency estimate updates is a multiple of T_m
  - Time offset between master→slave and slave→master messages may be initialized randomly or initialized with user-specified values
  - Time offset between master→slave and slave→master messages may remain constant over the simulation or vary over T_m by the relative frequency offset between master and slave, multiplied by T_m
    - Former requires that the master and slave send messages at the same rate
    - Latter corresponds to messages being sent at the free-running clock rates
  - Finite precision of clock is modeled
    - Granularity, in units of time, is supplied as input parameter

Parameters Common to All Simulation Cases

Leading the Next

- 10 hops
  - GM followed by 10 slave clocks, in chain
- Slave clock frequency tolerance = $\pm$ 100 ppm
- Inter-message time (T_m) = 1 ms
- Time between frequency offset updates = 10 ms (if frequency offset is estimated)
- Filter bandwidth = 10 Hz
- Filter gain peaking = 0.1 dB
- Simulation time step = 0.01 ms
  - Used small time step to ensure phase peaks were captured

Simulation Cases 1 and 2

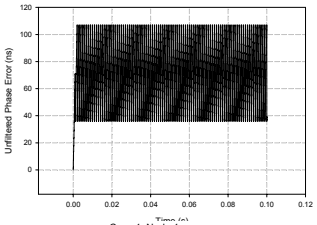
Leading the Next

- Assumptions
  - No clock phase noise
  - Granularity of clock = 0
  - No frequency adjustments (Case 1); Instantaneous frequency adjustments (Case 2)
  - Offset between master→slave and slave→master messages set to T_m at each node (deterministic and constant)
- Results (see plots on next slide)
  - With instantaneous phase adjustments (no filtering) and no frequency adjustments, steady-state peak-to-peak phase error can be large (tens of ns) and depends on frequency offsets
    - With 10 Hz filter and no frequency adjustments, steady-state peak-to-peak phase error is reduced to a few tenths of a ns
  - With instantaneous frequency adjustments, steady state peak-to-peak phase error is very small
    - Approximately 0.07 ns with no filtering
    - Approximately 0.00055 ns (0.55 ps) with filtering
    - With no clock noise and zero phase granularity, frequency offsets can be measured very accurately
  - Phase variation does not increase monotonically with number of clocks in chain

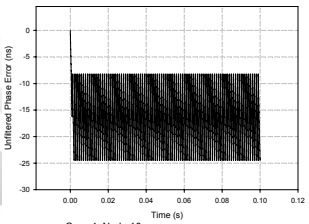
Simulation Case 1

Leading the Next

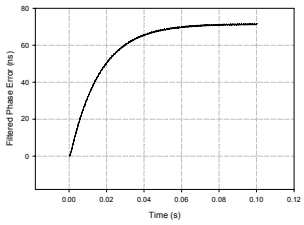
Case 1, Node 1
Instantaneous Phase Adjustments
No Frequency Adjustments



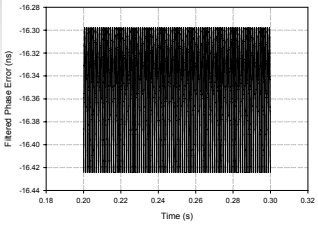
Case 1, Node 10
Instantaneous Phase Adjustments
No Frequency Adjustments



Case 1, Node 1
Filtered Phase Adjustments
No Frequency Adjustments
(Initial transient is included)



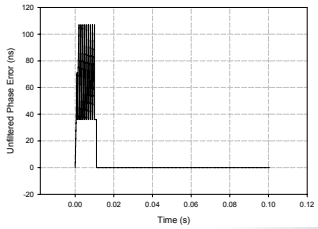
Case 1, Node 10
Filtered Phase Adjustments
No Frequency Adjustments
(Plot begins after initial transient has decayed)



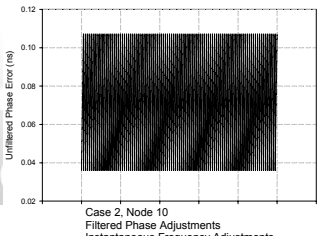
Simulation Case 2

Leading the Next

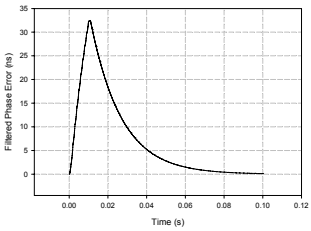
Case 2, Node 1
Instantaneous Phase Adjustments
Instantaneous Frequency Adjustments



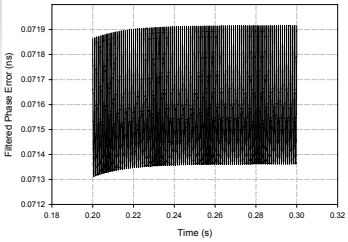
Case 2, Node 10
Instantaneous Phase Adjustments
Instantaneous Frequency Adjustments



Case 2, Node 1
Filtered Phase Adjustments
Instantaneous Frequency Adjustments



Case 2, Node 10
Filtered Phase Adjustments
Instantaneous Frequency Adjustments



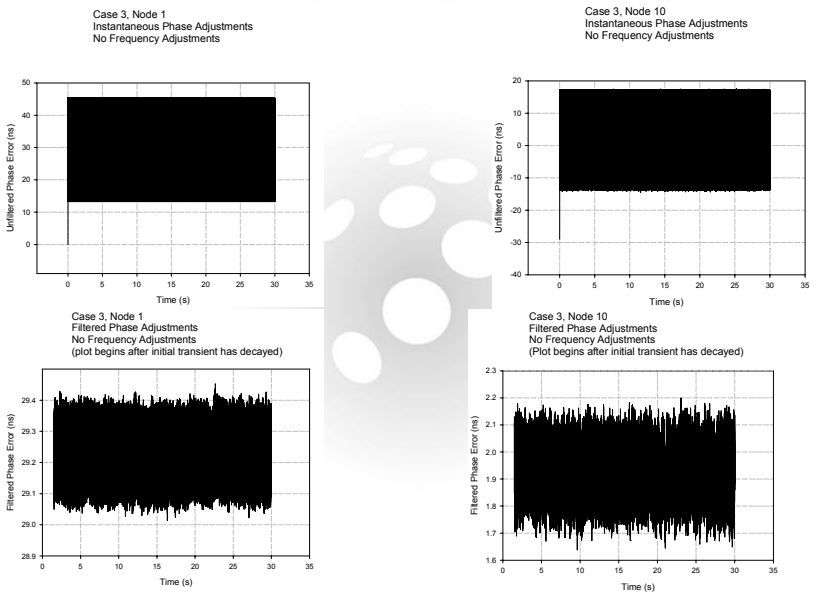
Simulation Cases 3 and 4

Leading the Next

- Assumptions
  - With clock phase noise (model described in Appendix)
  - Granularity of clock = 1 ns
  - No frequency adjustments (Case 3); Instantaneous frequency adjustments (Case 4)
  - Offset between master→slave and slave→master messages initialized randomly at each node
    - All nodes send messages at the same rate (offsets remain constant over simulation)
- Results (see plots on next slide)
  - With 10 Hz filter, MTIE is in roughly the same range with and without frequency adjustments at longer observation intervals
    - MTIE is slightly smaller with frequency
    - Maximum peak-to-peak phase variation is around 1 ns for both cases
  - Without filtering, MTIE ranges from approximately 15 – 60 ns without frequency adjustments and 1 – 4 ns with frequency adjustments
  - Phase variation does not increase monotonically with number of clocks in chain (in all cases)
  - Note that the results exhibit large statistical variability
    - Must run multiple, independent replications of the simulations to obtain confidence intervals for the results

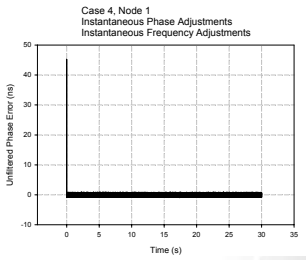
Simulation Case 3

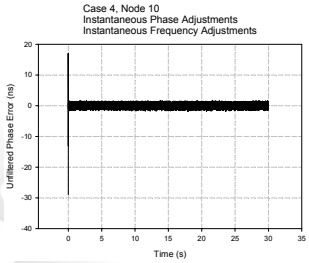
Leading the Next

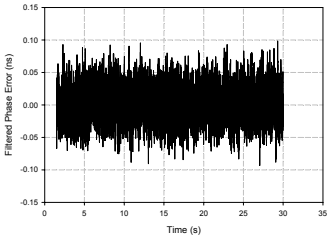


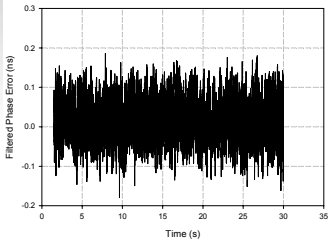
Simulation Case 4

Leading the Next



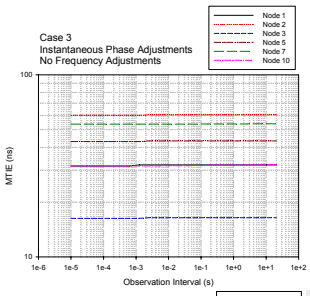


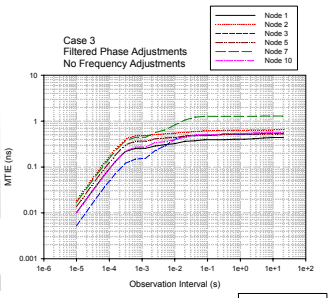


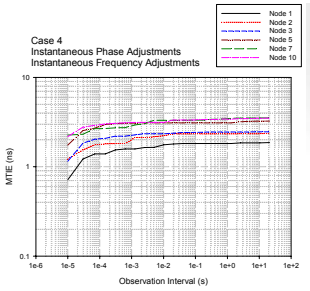


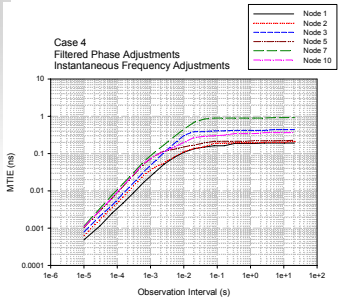
Simulation Cases 3 and 4

Leading the Next









Simulation Cases 5 and 6

Leading the Next

- Assumptions
  - With clock phase noise (model described in Appendix)
  - Granularity of clock = 1 ns
  - No frequency adjustments (Case 5); Instantaneous frequency adjustments (Case 6)
  - Offset between master→slave and slave→master messages initialized randomly at each node
    - All nodes send messages at local free-running clock rate (offsets vary over simulation)
- Results (see plots on following slides)
  - If frequency adjustments are not made, phase steps occur due to variation in time offset between master→slave and slave →master messages
    - This time offset results in a phase error equal to the size of the offset (in units of time) multiplied by the fractional frequency difference between the free-running master and slave clocks
    - As the time offset increases from 0 to T_m (or decreases from T_m to 0, phase offset changes
    - When the time offset reaches T_m (or 0) it jumps to 0 (or T_m) as one message "walks past" the other
    - This produces a step change in phase error equal to yT_m , where y is the relative frequency offset between the master and slave
      - –E.g., for $T_m = 0.001$ s and $y = 100$ ppm, the phase error jump is 100 ns

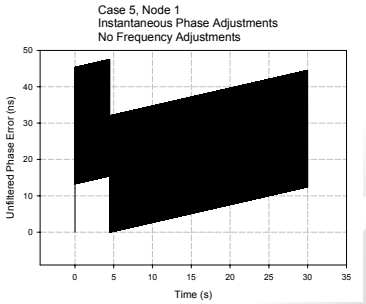
Simulation Cases 5 and 6

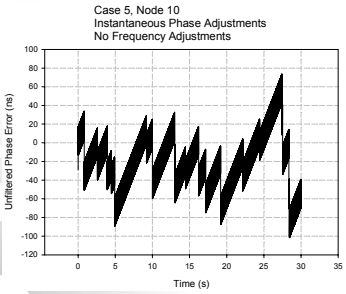
Leading the Next

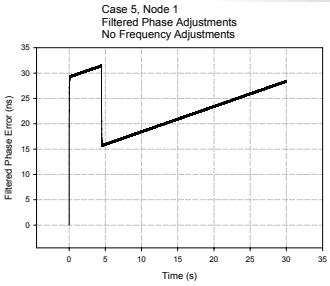
- Results (Cont.)
  - The 10 Hz filter removes the fast phase variation due to instantaneous phase adjustments, clock phase noise, and non-zero granularity; however, it cannot remove the phase variation due to variation in the time offset between the master→slave and slave→master messages as this variation is much slower
  - The effect does not occur when frequency adjustments are made because the error in phase correction due to the frequency offset between the nodes is corrected for
  - MTIE for the case with frequency adjustments is roughly the same as in the corresponding case where the master→slave and slave→master message time offset does not vary (Case 4)
  - Phase variation does not increase monotonically with number of clocks in chain (in all cases)
  - Note that the results exhibit large statistical variability
    - Must run multiple, independent replications of the simulations to obtain confidence intervals for the results

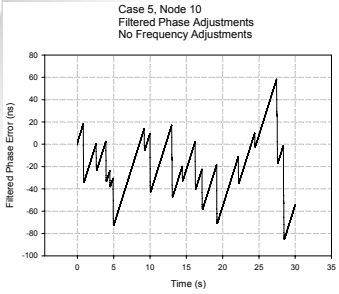
Simulation Case 5

Leading the Next



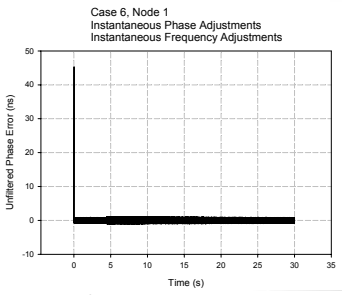


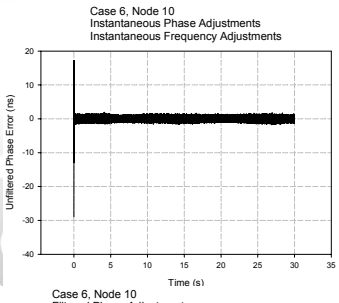


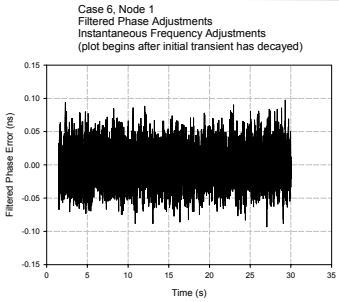


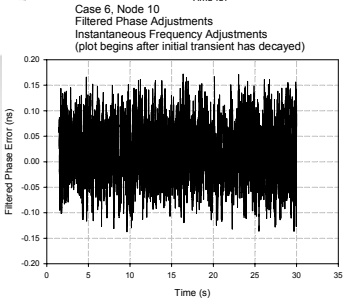
Simulation Case 6

Leading the Next



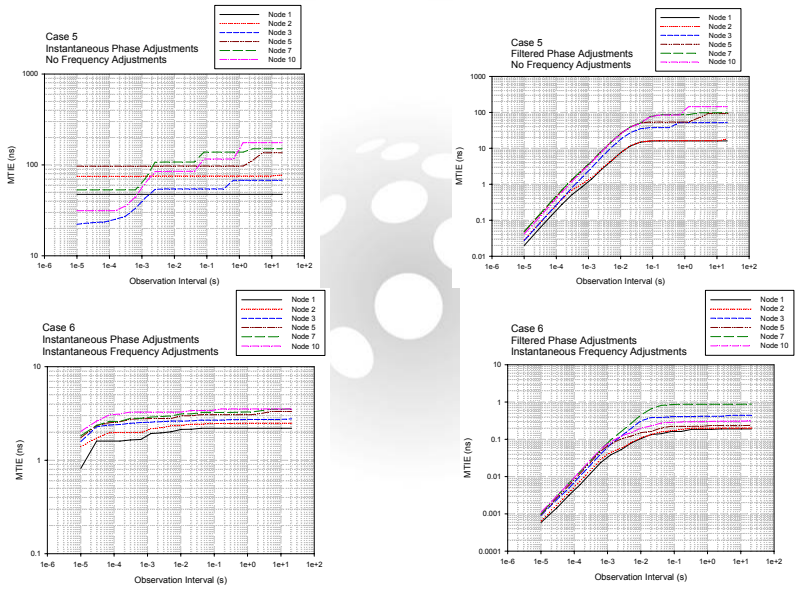






Simulation Cases 5 and 6

Leading the Next



Conclusions

Leading the Next

- In ideal case of no clock noise, zero phase granularity, and no variation in the time offset between the master→slave and slave→master messages , can achieve extremely small peak-to-peak phase variation in steady state
  - 0.07 ns with no filtering and frequency adjustments (Case 2, node 10)
  - 0.00055 ns with filtering and frequency adjustments (Case 2, node 10)
  - 0.12 ns with filtering and no frequency adjustments (Case 1, node 10)
- However, with clock noise (using the model of the appendix) and 1 ns phase granularity, peak-to-peak phase variation in steady state is larger
  - 1 – 4 ns with no filtering and frequency adjustments, whether or not time offset between the master→slave and slave→master messages vary
  - 0.2 – 1 ns with filtering and frequency adjustments, whether or not time offset between the master→slave and slave→master messages vary
  - 0.4 – 1.5 ns with filtering and no frequency adjustments if time offset between the master→slave and slave→master messages does not vary
  - 20 – 200 ns with filtering and no frequency adjustments if time offset between the master→slave and slave→master messages does vary

Conclusions

Leading the Next

- The cases with clock noise and 1 ns phase granularity indicate that MTIE masks for uncompressed digital video are exceeded if filtering is not done
  - This indicates that filtering is necessary, whether or not instantaneous frequency adjustments are made
  - While end-to-end digital audio masks are met, note that ResE gets only a budget allocation of the total (see [15] for digital audio reference models); also must consider statistical variability of the results (future work)
- The uncompressed digital video masks are slightly exceeded with 10 Hz, 0.1 dB filtering
  - Note that the masks apply to the end-to-end application
    - ResE gets only a budget allocation of the total
    - Get some additional phase variation (likely small) due to the finite granularity of the application time stamps relative to the synchronization signals described here
  - This means it is likely that the filter must have BW that is somewhat narrower than 10 Hz
- Results show that if instantaneous frequency adjustments are not made, must ensure that master→slave and slave→master messages are sent at nominally the same rate, to avoid variation of their time offset and resulting large phase variation for this case
- Note that only variations (2) – (4) (see slide 8) have been addressed here

Future Work

Leading the Next

- Analysis of additional parameter variations
  - Filter BW
  - Larger clock noise level
    - Choose level that bounds noise in oscillators expected to be used in ResE
  - Clock phase granularity
- Determination of statistical confidence intervals for MTIE (and possibly TDEV) by running multiple, independent replications a simulation case
- Analysis of other variations/choices (slide 8)

References

Leading the Next

1. Geoffrey M. Garner and Felix Feng, *Delay Variation Simulation Results for Transport of Time-Sensitive Traffic over Conventional Ethernet*, Samsung presentation at July, 2005 IEEE 802.3 ResE SG meeting, San Francisco, CA, July 18, 2005. Available via http://www.ieee802.org/3/re_study/public/index.html.
2. Geoffrey M. Garner, *End-to-End Jitter and Wander Requirements for ResE Applications*, Samsung presentation at May, 2005 IEEE 802.3 ResE SG meeting, Austin, TX, May 16, 2005. Available via http://www.ieee802.org/3/re_study/public/index.html.
3. Ralf Steinmetz, *Human Perception of Jitter and Media Synchronization*, IEEE JSAC, Vol. 14, No. 1, January, 1996, pp. 61 – 72.
4. *Residential Ethernet (RE) (a working paper)*, Draft 0.136, maintained by David V. James and based on work by him and other contributors, August 10, 2005. Available via http://www.ieee802.org/3/re_study/public/index.html
5. ITU-T Recommendation G.810, *Definitions and Terminology for Synchronization Networks*, ITU-T, Geneva, August, 1996, Corrigendum 1, November, 2001.
6. ITU-T Recommendation G.8251, *The Control of Jitter and Wander within the Optical Transport Network (OTN)*, ITU-T, Geneva, November, 2001, Amendment 1, June, 2002, Corrigendum 1, June, 2002.
7. Geoffrey Garner, *Jitter Analysis for Asynchronous Mapping of a Client Signal into an Och*, Lucent Contribution to ITU-T Q 11/15 Interim Meeting, Ottawa, ON, July, 2000.
8. *Phase Noise*, Vectron International, Application Note, available at http://www.vectron.com.

References

Leading the Next

9. *Jitter and Signal Noise in Frequency Sources*, Raltron, Application Note, available at http://www.raltron.com/
10. David W. Allan, Marc A. Weiss, and James L. Jespersen, *A Frequency Domain View of Time Domain Characterization of Clocks and Time and Frequency Distribution Systems*, Forty-Fifth Annual Symposium on Frequency Control, Los Angeles, CA, May 29 – 31, 1991, pp. 667 – 678.
11. Stefano Bregni, *Synchronization of Digital Telecommunications Networks*, Wiley, 2002.
12. J.A. Barnes and Stephen Jarvis, Jr., *Efficient Numerical and Analog Modeling of Flicker Noise Processes*, National Bureau of Standards, NBS Technical Note 604, June, 1971.
13. James A. Barnes and Charles A. Greenhall, *Large Sample Simulation of Flicker Noise*, 19th Annual Precise Time and Time Interval (PTTI) Applications Planning Meeting, December, 1987.
14. Giovanni Corsini and Roberto Saletti, *A $1/f^{\gamma}$ Power Spectrum Noise Sequence Generator*, IEEE Transactions on Instrumentation and Measurement, Vol. 37, No. 4, December, 1988, pp. 615 – 619.
15. Alexei Beliaev, *Latency Sensitive Application Examples*, Gibson Labs, part of *Residential Ethernet Tutorial*, IEEE 802.3 meeting, March, 2005.

Appendix I – Clock Noise Model

Leading the Next

- Clock phase noise may be modeled as a sum of random processes with power spectral density (PSD) of the form $Af^{-\alpha}$
  - In practice, the PSD has 3 terms (see [8] and [9])
    - α = 0, White Phase Modulation (WPM)
    - α = 1, Flicker Phase Modulation (FPM)
    - α = 3, Flicker Frequency Modulation (FFM)
  - Can write the PSD, $S_x(f)$ as

$$S_x(f) = \frac{A}{f^3} + \frac{B}{f} + C, \text{ where } S_x(f) \text{ has units of } \text{ns}^2/\text{Hz}$$

  - Often express as

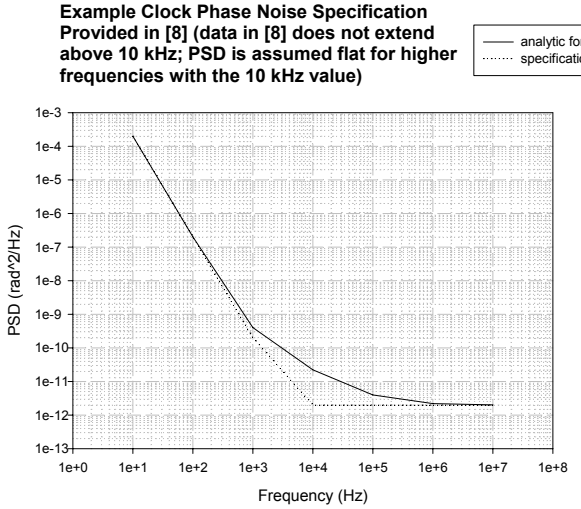
$$S_\phi(f) = (2\pi\nu_0)^2 S_x(f), \text{ where units of } S_\phi(f) \text{ are } \text{rad}^2/\text{Hz}$$

  - An example PSD specification is given in Figure 12 of [8], and reproduced on the next slide
    - Data in [8] is given in dBc/Hz; data has been converted to rad²/Hz
    - Data in [8] is given only for frequencies below 10 kHz; here, we assume the PSD is flat above 10 kHz
    - Dotted curve on the next slide is the converted data of [8]; solid line is a conservative fit of the above power law sum
  - The specifications for the individual products of [7] and [8] are below this example, at least for those products where phase noise specifications are provided

Appendix I – Clock Noise Model

Leading the Next

Example Clock Phase Noise Specification Provided in [8] (data in [8] does not extend above 10 kHz; PSD is assumed flat for higher frequencies with the 10 kHz value)



Note: Data in [8] is given in dBc/Hz; data has been converted to rad²/Hz

Appendix I - Clock Noise Model

Leading the Next

- Another measure for clock noise, which is more convenient because it is a time domain parameter, is Time Variance (TVAR)
  - Time Deviation (TDEV) is the square root of TVAR
- TVAR is 1/6 times the expectation of the square of the second difference of the phase errror averaged over an interval

$$\mathrm{TVAR}(\tau) = \frac{1}{6} E\left[\left(\Delta^2 \bar{x}\right)^2\right]$$

where $E[\cdot]$ denotes expectation,
$\bar{x}$ denotes average over the integration time τ,
and Δ^2 denotes second difference

- TVAR may be estimated from measured or simulated data using [5]

$$\mathrm{TVAR}(n\tau_0) = \frac{1}{6n^2(N-3n+1)} \sum_{j=1}^{N-3n+1} \left[\sum_{i=j}^{n+j-1} \left(x_{i+2n} - 2x_{i+n} + x_i\right) \right]^2, \quad \mathrm{n} = 1, 2, \ldots, \text{integer part}(N/3)$$

where τ_0 is the sampling interval and $\tau = N\tau_0$

Appendix I - Clock Noise Model

Leading the Next

- TVAR is equal to $\tau^2/3$ multiplied by the Modified Allan Variance
- For power-law noises with PSD proportional to $f^{-\alpha}$, TVAR is proportional to τ^{β}, where $\beta = \alpha - 1$
- The magnitude of TVAR may be related to the magnitude of PSD for power-law noises; see [10] and [11] for details
  - FFM

$$S_x(f) = \frac{A}{f^3} \qquad \mathrm{TVAR}(\tau) = \frac{(2\pi)^2 9 \ln 2}{20} A \tau^2$$

  - FPM (result is from [10]; a more exact expression is given in [11])

$$S_x(f) = \frac{B}{f} \qquad \mathrm{TVAR}(\tau) = \frac{3.37}{3} B$$

  - WPM

$$S_x(f) = C \qquad \mathrm{TVAR}(\tau) = \frac{\tau_0 f_h}{\tau} C$$

f_h = noise bandwidth

Appendix I – Clock Noise Model

Leading the Next

- Simulation of WPM
  - WPM is simulated as a sequence of independent, identically distributed random samples
  - Noise distribution is taken as Gaussian with zero mean
  - Variance and sampling time determine TDEV level
    - Choose variance such that, with given sampling time, the computed TDEV from a sample history is close to value obtained from above relation between TDEV and PSD
      - Assume noise bandwidth is equal to line rate (100 MHz)
- Simulation of FPM
  - FPM is simulated by passing a sequence of independent, identically distributed random samples through a Barnes/Jarvis filter [12] – [14]
    - If white noise is input to a filter with frequency response $H(f) = f^{-1/2}$, the output is a random process with PSD proportional to $1/f$
    - The Barnes/Jarvis filter approximates an $f^{-1/2}$ frequency response using a bank of lead/lag filters
      - The actual frequency response of this filter is a "staircase"
      - The spacings of the poles and zeros are chosen such that the average slope is –10 dB/decade

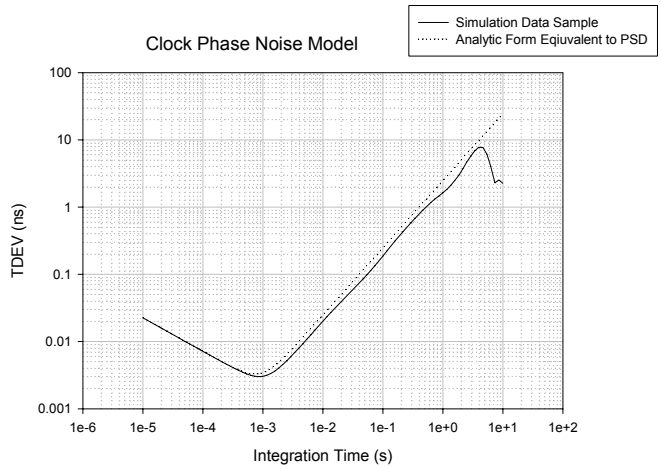
Appendix I – Clock Noise Model

Leading the Next

- Simulation of FPM (Cont.)
  - Noise distribution is taken as Gaussian with zero mean
  - Variance determines TDEV level
    - Choose variance such that the computed TDEV from a sample history is close to value obtained from above relation between TDEV and PSD
- Simulation of FFM
  - Input a sequence of independent, identically distributed random samples through a Barnes/Jarvis filter followed by an integrator (accumulator)
  - Noise distribution is taken as Gaussian with zero mean
  - Variance determines TDEV level
    - Choose variance such that the computed TDEV from a sample history is close to value obtained from above relation between TDEV and PSD
- Next slide shows TDEV for simulated data sample (10^{-5} s time step) and analytic form equivalent to PSD (solid curve on slide 35)

Appendix I – Clock Noise Model

Leading the Next



Appendix II – Definition of MTIE

Leading the Next

- Jitter and wander requirements can be expressed in terms of Maximum Time Interval Error (MTIE) masks
- MTIE is peak-to-peak phase variation for a specified observation interval, expressed as a function of the observation interval
  - An estimate of MTIE may be computed by (see [5])

$$\mathrm{MTIE}(n\tau_0) \cong \max_{1\le k\le N-n}\left\{\max_{k\le i\le k+n} x(i) - \min_{k\le i\le k+n} x(i)\right\}, \quad n = 1, 2, ..., N-1$$

where τ_0 is the sampling interval, $n\tau_0$ is the observation interval, $x(i)$ is the ith phase sample, and N is the number of phase samples ($N\tau_0$ is the measurement interval)

- The derivation of the MTIE masks on slide 6 from the jitter and wander requirements is given in [2]

Agilent Technologies

Rotational Synchronization via IEEE1588

Abstract

IEEE1588 is a standard that enables the precise synchronization of clocks in measurement & control systems implemented with network communication. This paper will describe the application of this Precision Clock Synchronization concept on establishing rotational synchronization between two Brushless Direct Current (BLDC) motors on two different nodes through Local Area Network (LAN).

Conventionally, the control point used by motor controller is speed, whereas the motor control systems attempt to maintain the speed through servo looping. However, when two motor control systems run separately, there is no mechanical rotation linkage; in other words, they are not run on the same phase angle. Therefore this application is to apply phase angle control on the motor control system. Phase angle of motor is a parameter reference to time of rotation. These two BLDC motors are being controlled separately by two motor control systems and these two systems are then being governed on its phase by different IEEE1588 nodes that are connected to the LAN. Based on synchronized clock and time over these two nodes, when both motors are running at the same phase angle, referring to same time base, the motors are thus rotationally synchronized.

This method of rotational synchronization via the IEEE1588 precision clock synchronization demonstrates machine control and automation application where the IEEE1588 standard can come into picture. This paper will discuss the methodology used to derive the phase of a motor, analyze the servo control loop used, the problems met and lessons learnt.

1.0 Objective

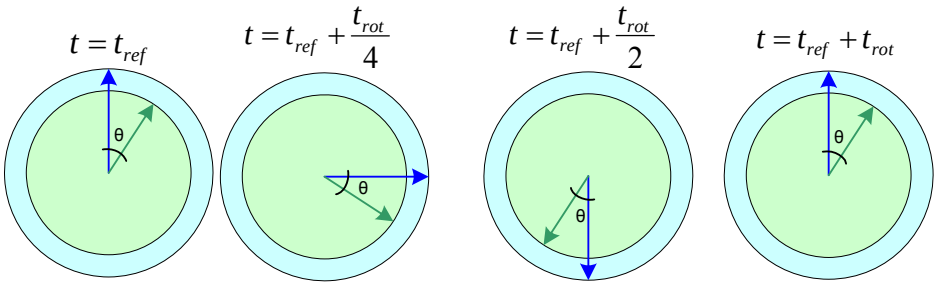
1.0 Establish a method to carry out rotational synchronization of two BLDC motors across Ethernet based on IEEE1588;
1.1 Evaluating of the accuracy of the rotational synchronization;
1.2 Articulate the advantages and disadvantages of this methodology;
1.3 Proposal for further research and improvement.

2.0 Introduction

IEEE1588 Time synchronization defines method to synchronize clocks for different systems at sub-microsecond over an Ethernet network and this allows precise time for these systems. This forms the basis of Real Time Application.

2.1 Rotational Synchronization

Rotational synchronization is defined as 2 or more motors, each rotating at the certain phase angle, mechanically at the encoder index point. Each of these motors will have to be well-controlled to achieve the same rotation period, t_{rot}, to the reference of a same starting time, t_{ref}, with a optional offset phase angle, θ, that is being maintained between these motors over the period of the synchronization.



To achieve Rotational Sync, both motor systems need to be referring to the same time base and well-controlled phase angle. This paper describes the methodology used to derive phase angle of a motor, how 1588 Clock Sync can lead to rotational sync and verification of the measurement.

3.0 Block diagram

3.1 Overview Block Diagram

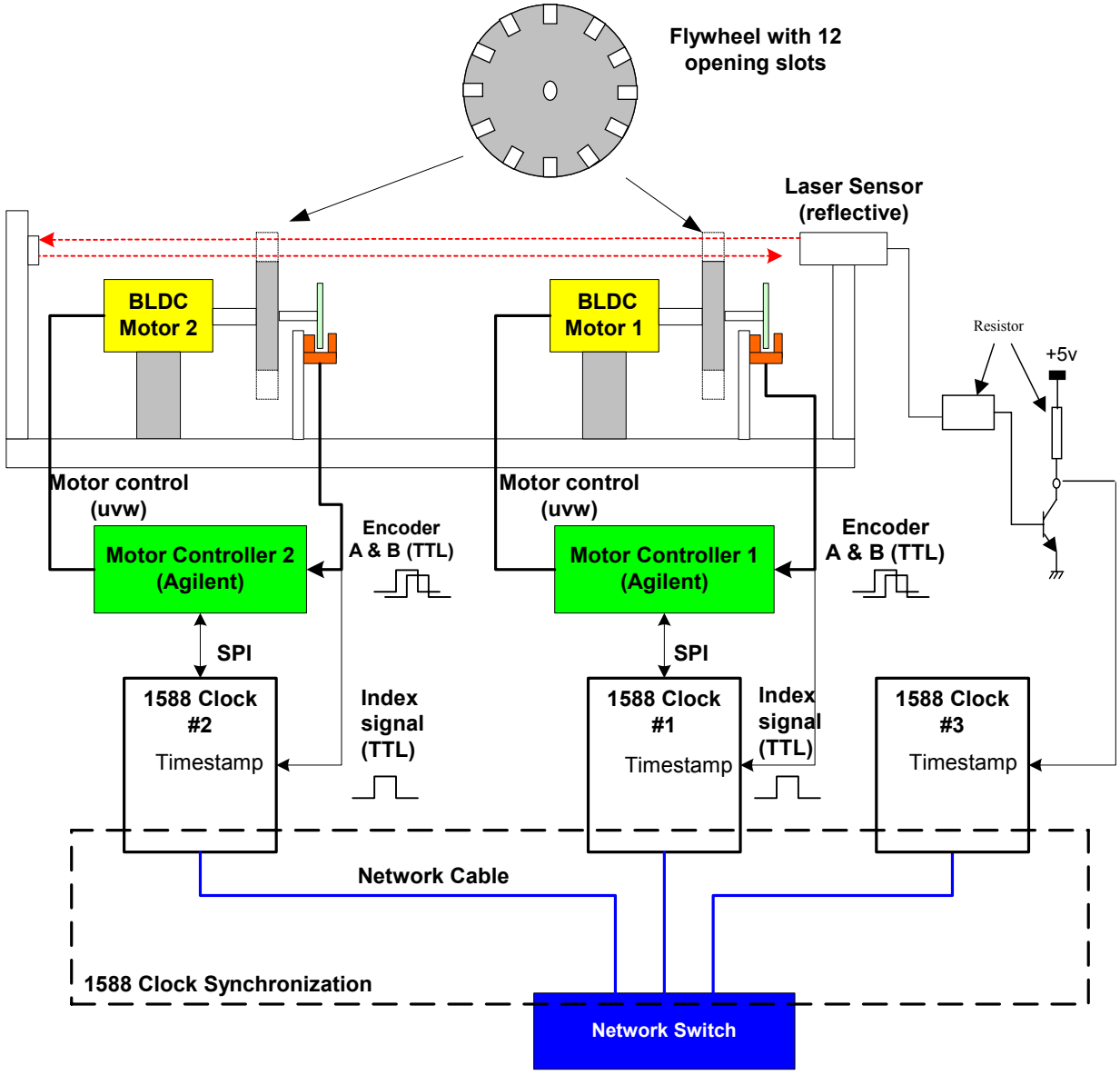


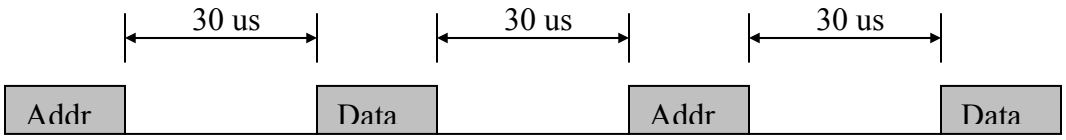
Figure 1: Overall block diagram of Demo Setup

4.0 Motor Controller –SPI commands

For Agilent motor controller, it uses SPI (**Serial Peripheral Interface**) protocol for communication.

4.1 Timing Diagram

The interval time between each address sending and data sending is minimum 30us.



4.2 Motion Control commands

Register commands	Address	Data
write to Command Velocity MSB	64	0
write to Command Velocity LSB	63	0

} Speed value

Table 1. Motor Controller speed commands

For summary, to control the speed of the BLDC motor, it takes 2 commands to write to the motor controller registers of speed (LSB and MSB respectively).

4.3 Conversion Speed to Motor Control Register value

By manipulating the Command Velocity register value, motor will rotate in different speed accordingly. Data has been collected from the register value written against rotation speed (on Channel A frequency)

4.3.1 Measurement of Speed (Radian/second)

The resolution of the encoder attached is **500CPR** (count per revolution). Then, the speed of rotation in frequency, which equal to round per second is: -

$$Frequency(Hz) = \frac{ChannelA(Hz)}{500}$$

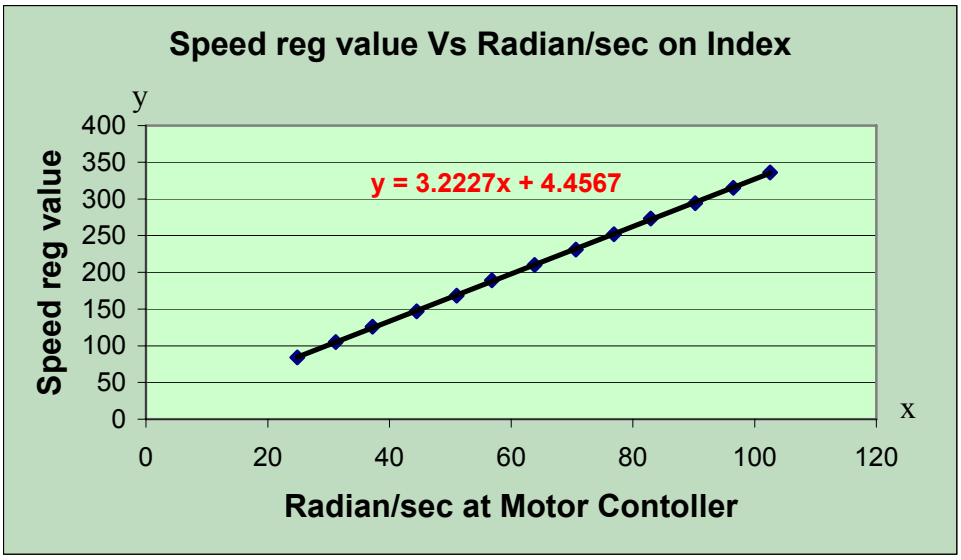
Then, speed in radian/second is: -

$$Speed(rad/s) = Frequency(Hz) \bullet 2\pi(rad)$$

Motor Controller register value			Feedback on Encoder				
(MSB)	(LSB)	Value	Channel A (Hz)	Index (Hz)	Speed (rpm)	radian/s	rpm/value
0x00	0x15	21	420	0.84	50.40	5.28	0.25
0x00	0x2A	42	925	1.85	111.00	11.62	0.28
0x00	0x3F	63	1450	2.90	174.00	18.22	0.29

0x00	0x54	84	1980	3.96	237.60	24.88	0.30
0x00	0x69	105	2480	4.96	297.60	31.16	0.30
0x00	0x7E	126	2960	5.92	355.20	37.20	0.30
0x00	0x93	147	3540	7.08	424.80	44.48	0.30
0x00	0xA8	168	4060	8.12	487.20	51.02	0.30
0x00	0xBD	189	4520	9.04	542.40	56.80	0.30
0x00	0xD2	210	5080	10.16	609.60	63.84	0.30
0x00	0xE7	231	5620	11.24	674.40	70.62	0.31
0x00	0xFC	252	6120	12.24	734.40	76.91	0.31
0x01	0x11	273	6600	13.20	792.00	82.94	0.30
0x01	0x26	294	7180	14.36	861.60	90.23	0.31
0x01	0x3B	315	7680	15.36	921.60	96.51	0.31
0x01	0x50	336	8160	16.32	979.20	102.54	0.31

Table 2. Speed registers value Vs speed



Graph 1. Speed register value Vs Radian/sec

Velocity is more stable starting from reg value of 84 onwards (200 rpm and above).
The formula of the relationship between register value and speed will be used to convert the desired rotation speed in SI unit (Radian/sec) to required register value.

5.0 TaoMaster Demo Application Software

5.1 Software structure

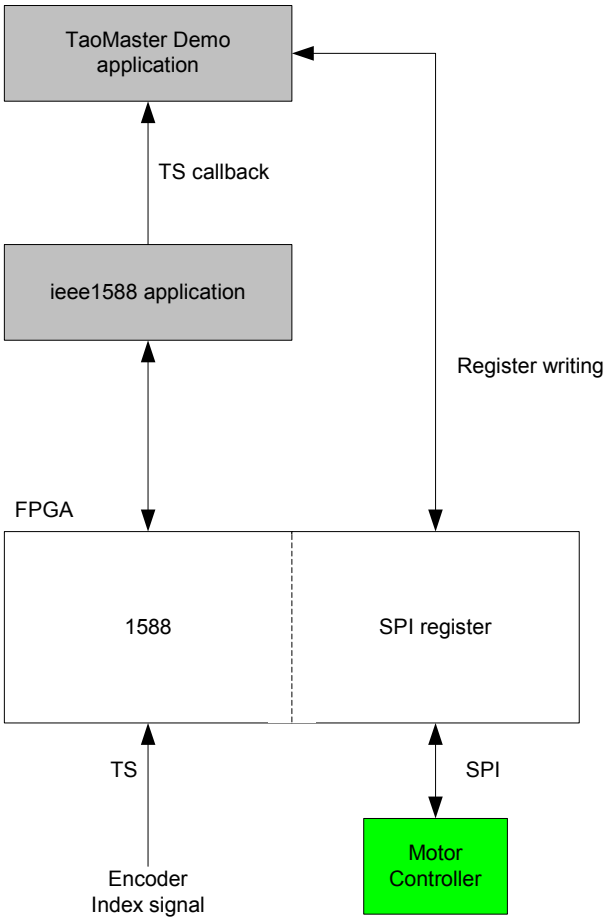


Figure 2: Overall Software structure

The TaoMaster demo application will be on a separate application that run on top and have dependency on existing ieee1588 application on the 1588 node. It will interface with the ieee1588 for the Timestamp (TS) callback interrupt and register writing for the all the dedicated SPI register at the FPGA for motor controlling.

TS interrupt will be used in two purposes: -

a. Calculate rotation speed – the velocity of the motor will the average time difference between TSs.
b. Calculate phase offset – the phase offset is the degree of angle offset the TS from a reference time.

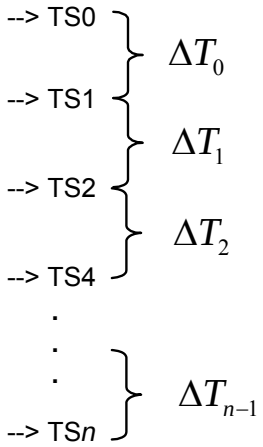
With the time already synchronized between all the 1588 nodes, by controlling the rotation speed and phase offset will hence get the motor in sync.

5.2 Principle of Software Implementation Concept

5.2.1 Rate calculation concept

Rotation rate (velocity) is calculated based on the TS input to which connected to the encoder Index channel. As index pulse will triggered once every rotation, therefore the period between every two consecutive TS represent the time taken for one complete rotation.

The period between TS is calculated in the average time in a set of ***n*** period of time collected: -



$$Average, T_r(\sec) = \frac{\sum_{a=0}^{n-1} \Delta T_a}{n-1}$$

$$Rotation, V_r(rad/s) = \frac{1}{T_r} \bullet 2\pi$$

Figure 3: Rotation rate calculation

Then rotation speed is calculated base on this average of period between ***n*** TS in radian/sec.

Method 1 – Ring element

Using a number of elements, each holds a set of record TS time. This entire element is link together to form a set of ring.

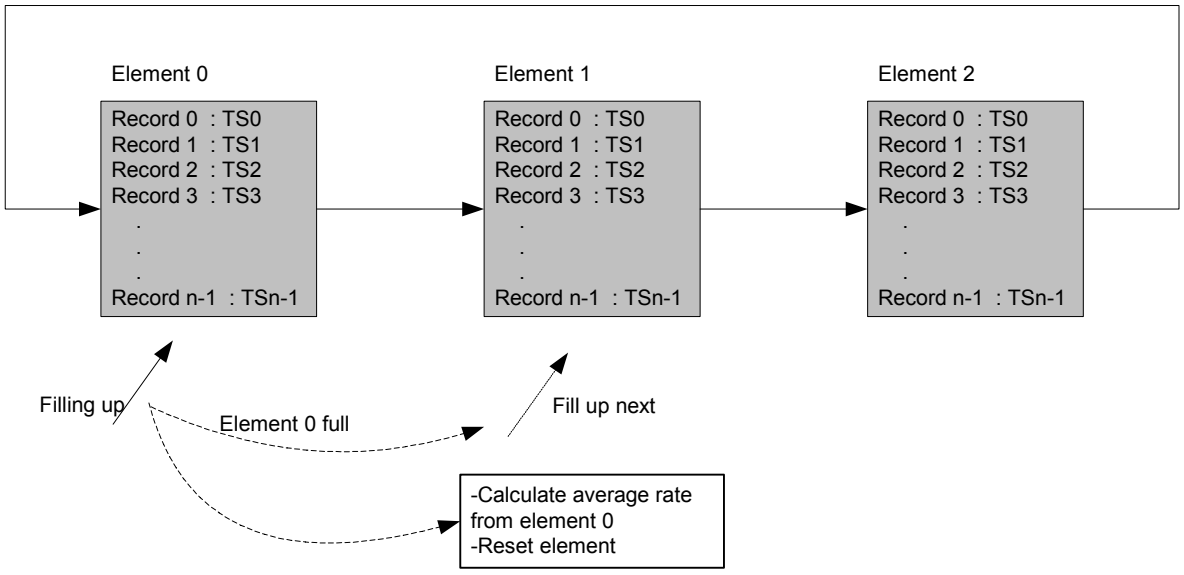


Figure 4: Ring Element

At first, the TS interrupt will start filling up the record list in element 0 in **ISR** routine. When the record list for element 0 is full, the next TS will be filled up in next element and at the same time wake up the task-level **thread processing** to calculate the average rate from the ***n*** record list, then clear the record list.
The process then continues until when the last element is filled up, it will move back to element 0 to start all over again.

Method 2 – Running Average
In this method, only an element is used where each TS time is recorded and the delta time is being added into a running total value. When the record is full, the next TS will start filling again at first record, record 0 and so on.

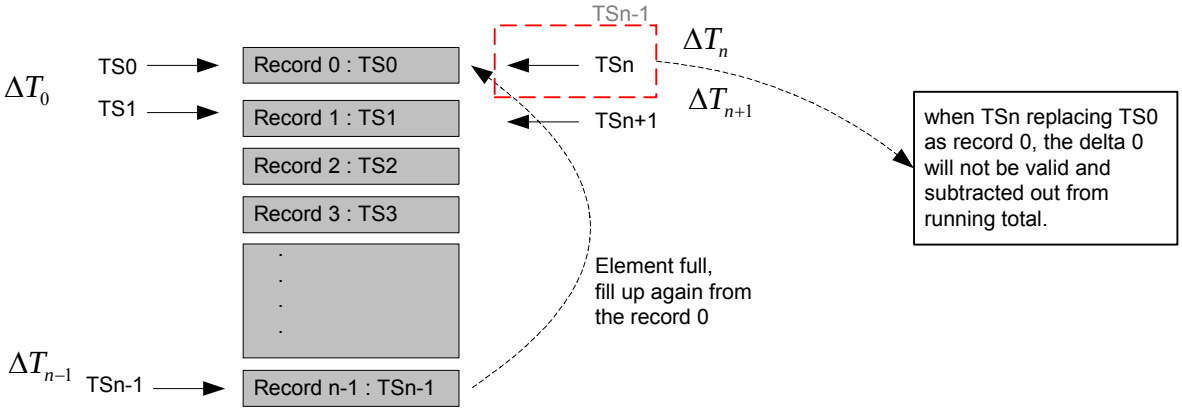


Figure 5: Running Average

The oldest delta is then subtracted out from the running total value. In summary, the running total will only maintain a fix set of delta time where the **oldest delta time will be subtracted in order to add a new delta time**, hence is called ***Running Total***. The period can then be calculated by dividing the Running Total to the number of set delta, which wills constant ***n*** after the record is first full.

5.2.2 Phase calculation concept

Phase is the measurement of offset of rotation index point, in angle, to the desired occurrence time from the reference time, t_{ref} +(t_{rot} x n) or $t_{index(n)}$, where as n is the rotation cycle time.

Phase is measured by comparing the TS input time to a reference time, the offset degree in radian of the TS from the reference point with the desired rotation period.

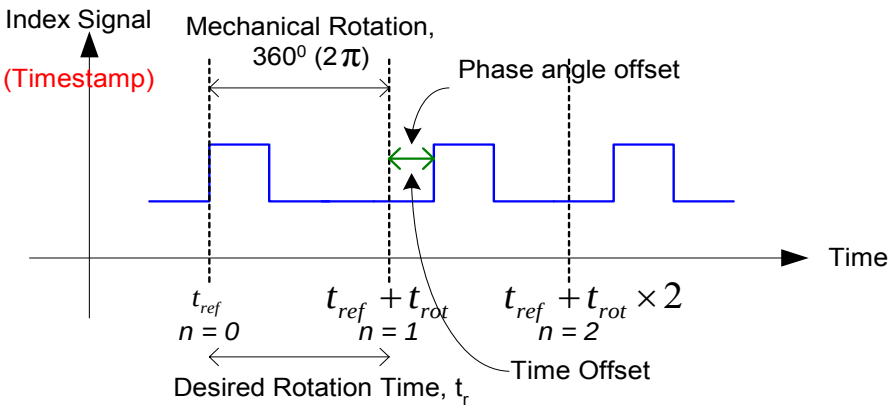


Figure 6: Phase angle Concept

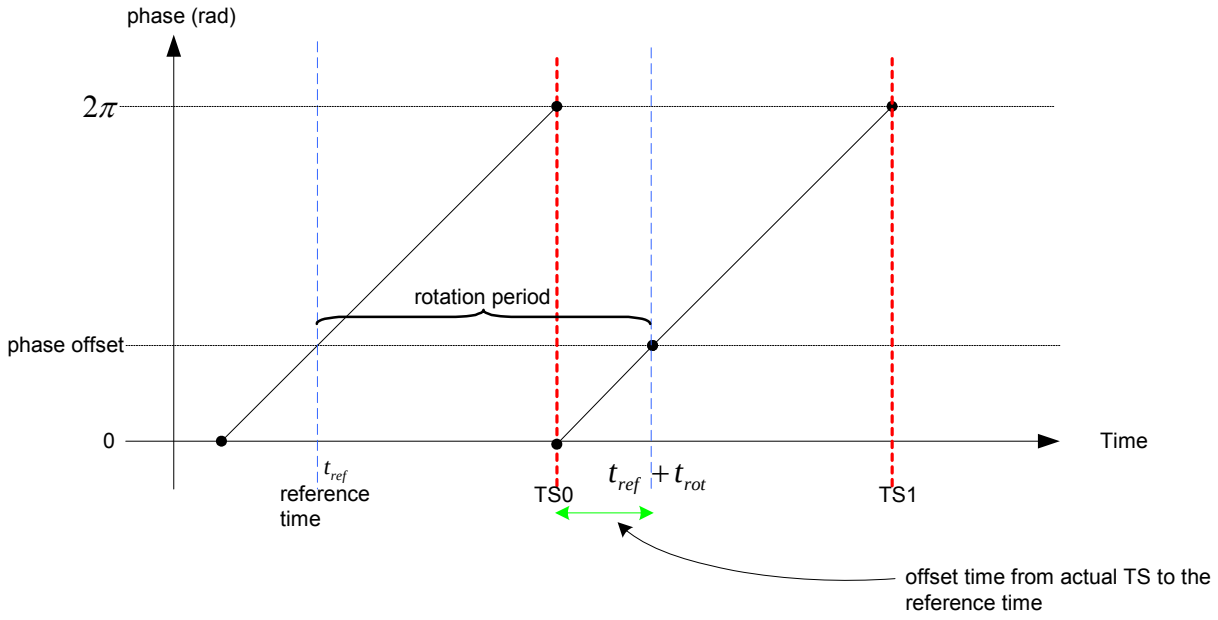


Figure 7: Rotation phase offset concept

On Figure 7, every rotation will start of 0 radian to 2π radian, referring to the index channel. With a reference time set, the offset time from actual TS to the **new reference time** from rotation rate is then can be calculated in ratio to the angle.

A reference time chart is to generate latest reference time base on desired the start time and desired period (converted from desired rate)

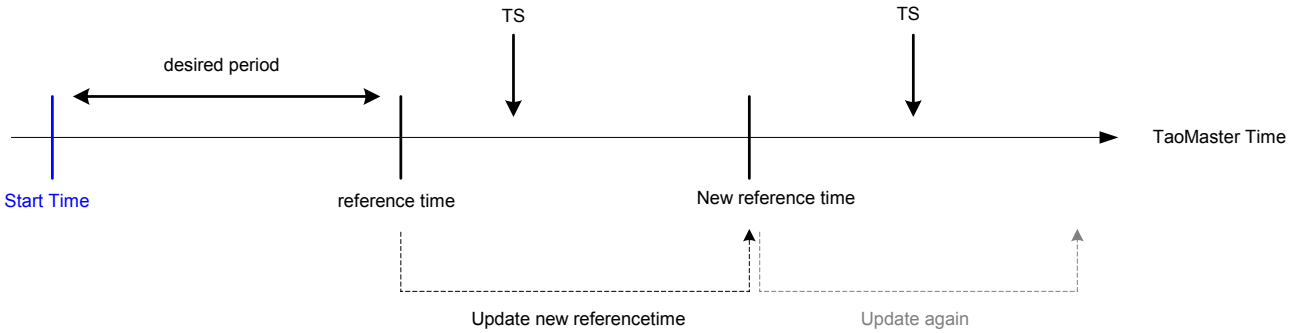


Figure 8: Reference Time Chart

As of the relation ship between the period and the angle is in linear, therefore by applying the ratio theorem, the phase can be calculated as per figure 9.
The phase will be calculated in two regions, separated by halfway point of the desired rotation period. This is to choose the closet reference time as the base of the calculation. As a result, the region that is less than half will have positive value and the other region yield negative phase value.

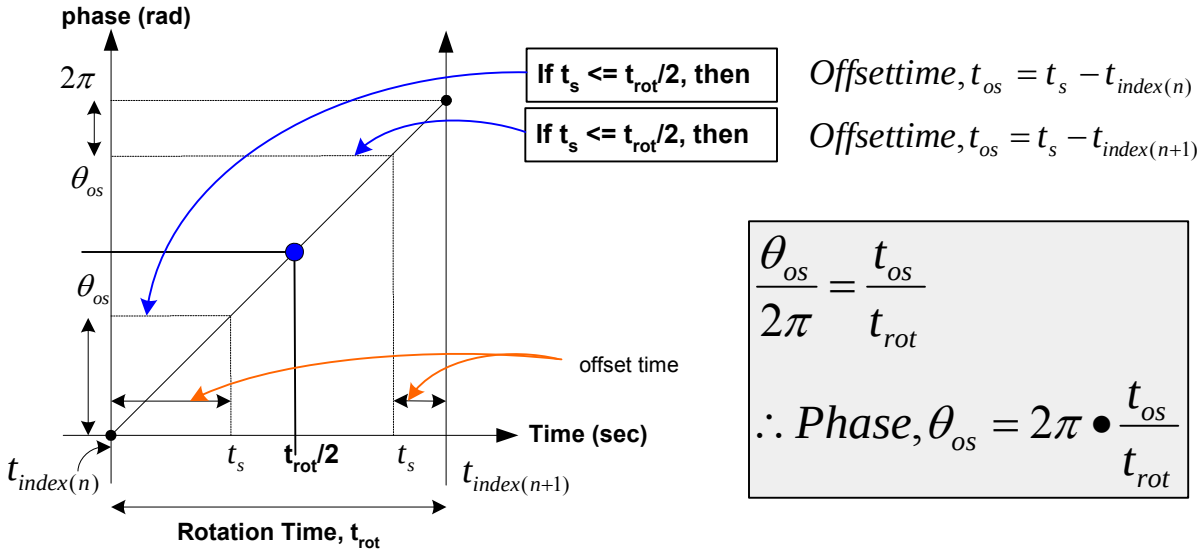


Figure 9: Rotation phase offset calculation

5.2.3 Controlling rate & phase

Overview of the structure of the control structure: -

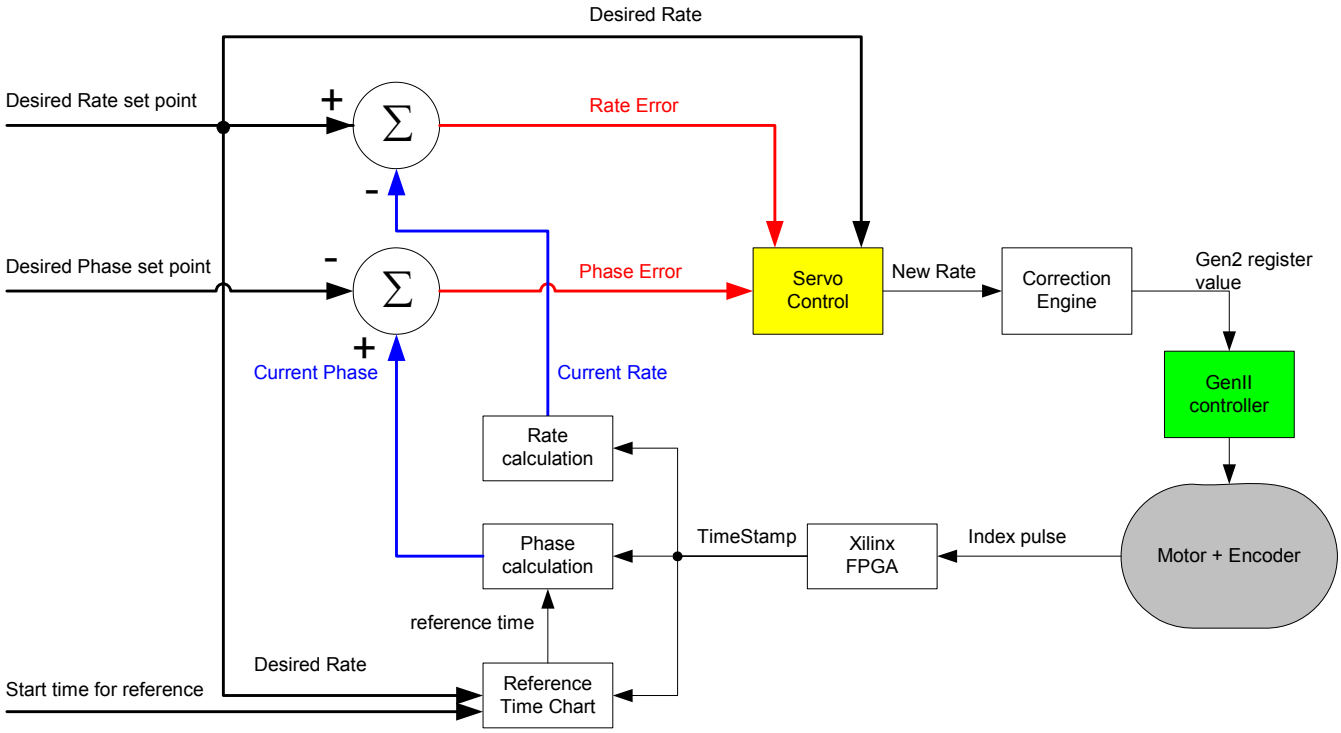


Figure 10: Overview of control loop

Three inputs to the control loop are: -

a) Desired rate (rotation speed)
b) Desired phase (angle in radian)
c) Start time reference for phase

Rate or phase error is calculated from subtracting the current measured value from the desired set points.

5.2.4 Servo Control Loop

To adjust the phase angle, a simple servo control loop (PI mode) can be applied into controlling the phase angle. The measured phase angle value of positive indicates that the index point is lagging behind from the closet desired time, which is slower. Then the control loop needs to speed up the rotation. In the other way, if the value is negative, which the index point is further in front, which is the speed is faster. The control then needs to slow down to rotation

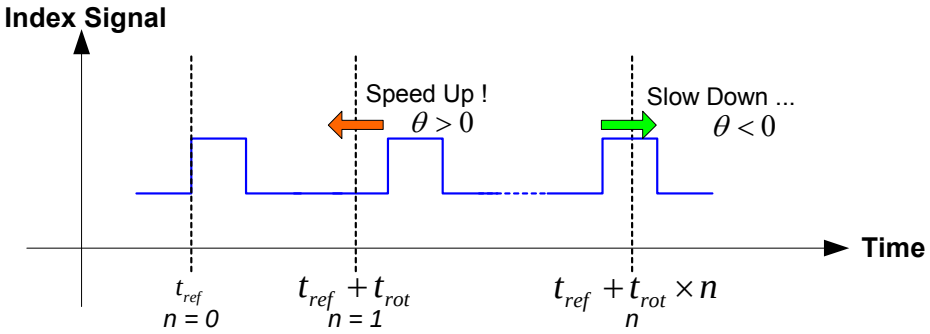


Figure 11: Controlling phase angle

Control Calculation module will take the phase error together with the desired rate to calculate a new Motor rate to send to motor controller, after through the correction engine. Hence when the new rate takes effect, the new calculated rate and phase will be feedback again for further looping control.

Control Calculation will be implemented with **PI** mode (proportional-integral). The sample rate of the servo, which is the interval number of index pulses for every new control value, is being calculated and executed.

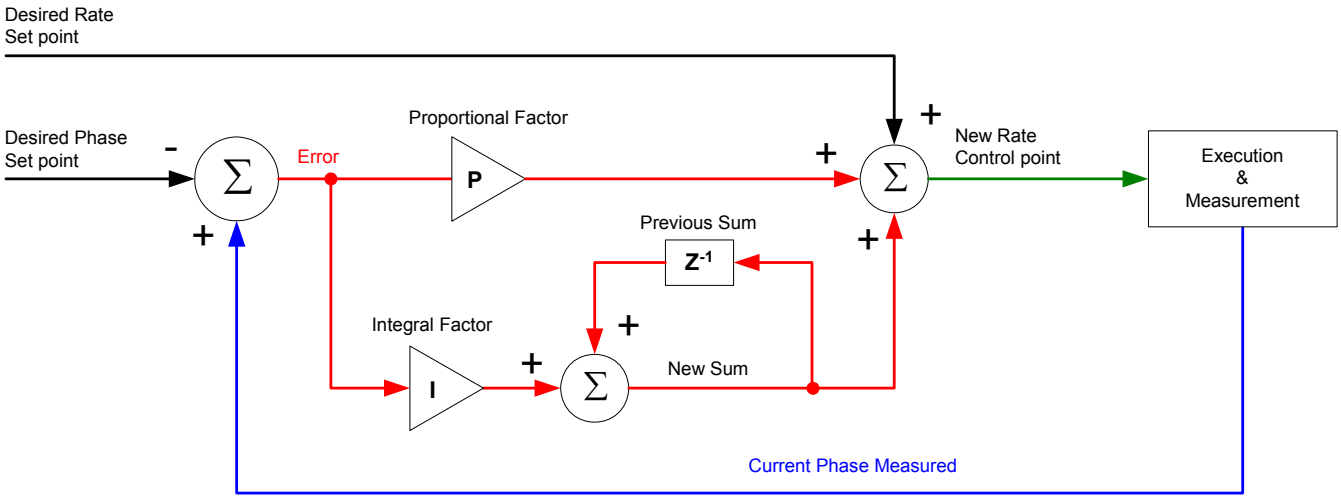


Figure 12: Control Calculation (PI mode)

6.0 Results & Observation

6.1 Methodology of Synchronization Accuracy Measurements

To measure the rotational synchronization between the two motors, each motor is attached with a flywheel with identical slotted holes and a reflective laser sensor between them at the slots. The laser beam will be turn on whenever the both motors' slots are lined up and the laser beam can pass through. The output of the sensor is being measured by another 1588 node as in speed.

The slot opening is about 3° each and there are 12 slots on each flywheel. Therefore, when both motors are fully synchronized, both flywheels are consistently lined up slot-to-slot enabling the laser beam to pass through at all 12 times per rotation. The speed measured by the laser beam will be 12 times the speed of the flywheel.
In this case, the speed would be expected at 480 rad/s (12 x 40 rad/s). Its ability to measure at 480 rad/s at all times indicates that both flywheels are rotational synchronized.

6.2 First result

Below parameter value has been used: -

Motor 1
Desired Rate = 40 rad/s
Desired Phase = 6.266 rad (≈ 359°)*
Reference Time = 20050315000000 (00:00 at 15 March 2005)

Proportional value = 0.15
Integral value = 0.01
Control Sample rate = 5

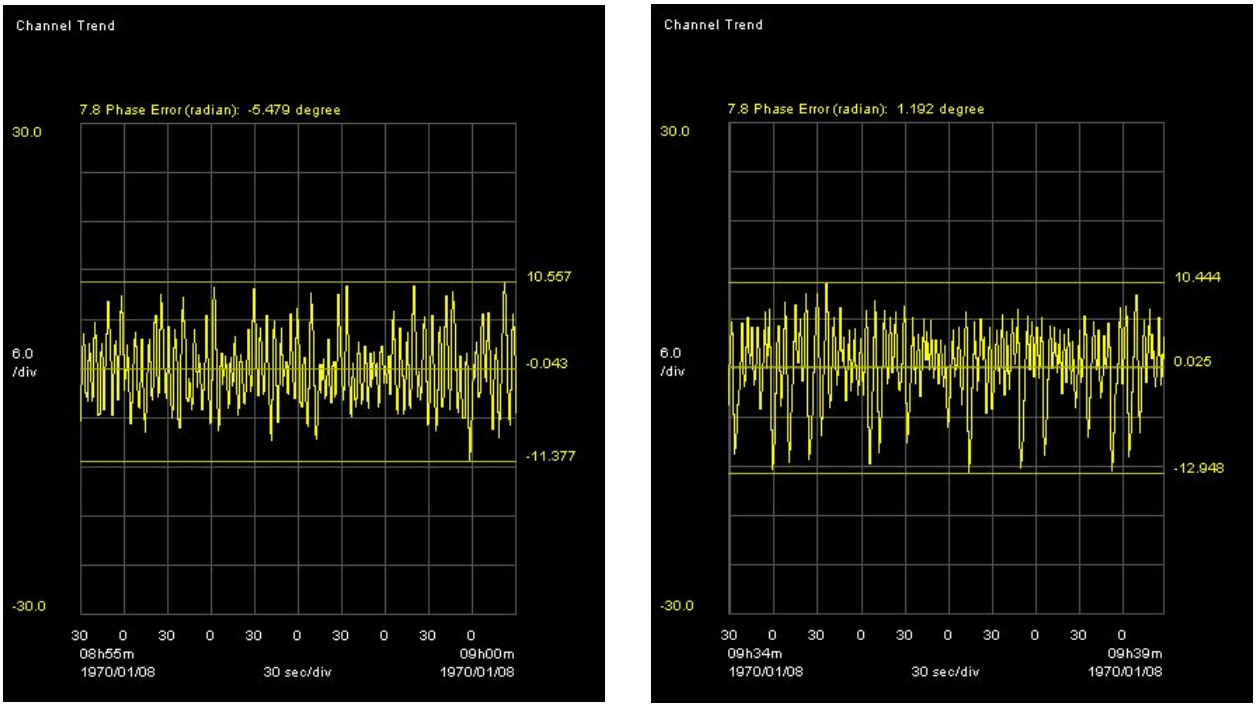
Motor 2
Desired Rate = 40 rad/s
Desired Phase = 0 rad (= 0°)
Reference Time = 20050315000000 (00:00 at 15 March 2005)

Proportional value = 0.15
Integral value = 0.01
Control Sample rate = 5

Note*: 1 degree offset is because of the flywheel mounting offset from motor 1 compare to motor 2 flywheel mounting

6.2.1 Phase control Accuracy

By implementing the PI servo loop to control the phase, both motor achieve the accuracy of the phase angle of about **±12°**.



Motor 1 Motor 2

Figure 13: Phase Control Accuracy at using 500CPR encoder

6.2.2 Rotation Synchronization result

On the laser measurement plot, we notice an uneven measurement of speed,

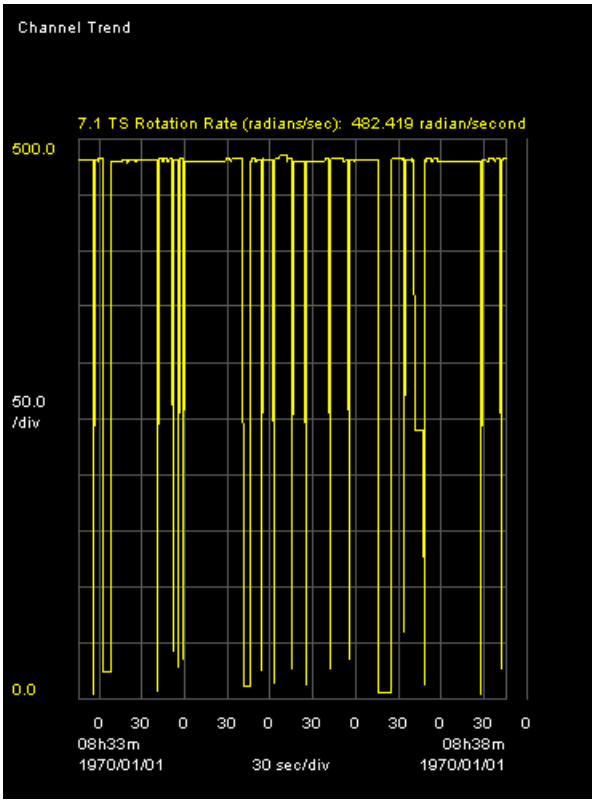


Figure 14: Rotation Synchronization at using 500CPR encoder

6.2.3 Observation

The result shows that the rotational synchronization is only achieved occasionally, where the speed measured at 480 rad/s is rare. In the Rate vs. Time plotting (referring to figure 14), the speeds measured by the sensor are varying between 480 to somewhere near zero.
This is because that the both flywheels are just at sometime rotationally synchronized, so then the speed measured is at 480 rad/s. But when is out, then the speed will dropped to close to zero (as the rate is calculated by running average).

The main reason behind for not rotational sync is that the phase control accuracy at only of about ±12°, whereas the opening slot is at 3°. Therefore whenever the difference of phase between both motors runs more than 3°, then slots are not lined up together and block the laser beam.

6.2.4 Investigation

The fundamental problem is too course of a control over the motor controller rotation rate. The phase control is using servo loop that calculated the new motor speed to be feed into the motor controller speed register value. As for the current setup, the relationship between the speed and the register value (that being obtained at section 4.4) is: -

$$register = 3.2227 \bullet Speed + 4.4567$$

Register is the actual motor controller register value truncated to a 16-bit value number and speed is the desired rotation rate in radians/sec.

In other words, Speed to Register value ratio is: -

$$Speed = 0.3103 \bullet register - 1.3829$$

Register value will change only about every 0.3 rad/s step changes. This creates a “stair steps” where the changes of speed of value smaller than 0.3 rad/s will be truncated into the same register value, which are roughly about 17°.

Thus, when controlling the speed for phase, it does not responsive enough where small changes will not be taken effect to the motor as of already being truncated to the same value. In other words, when servo loop calculation issues a new rate that is being truncated to the same register value, then the phase error will not be tackled and accumulated until the new rate calculated is able to trigger to the next register value.
The main reason of this limitation is that the Encoder resolution used is only 500CPR and the motor controller register value manipulation is based on the encoder resolution.

6.3 Proposal of Solution

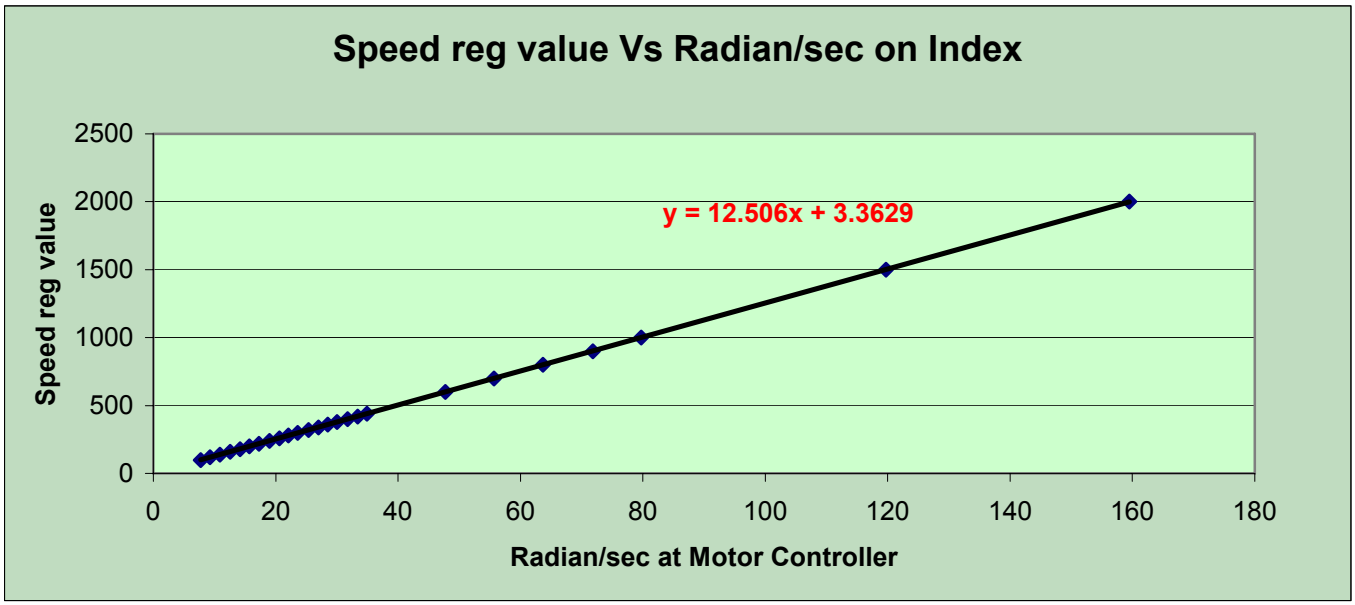
In viewing that the low phase control accuracy (±15°) is due to limitation of the speed register value controllability; the proposal is to use another higher accuracy encoder that will give a greater control over the speed variation.

A 2000CPR encoder is being chosen as a replacement for the original 500CPR encoder to minimize the “stair steps” effect on the truncation of speed rate to register value.

The speed vs. register value was measured as per the same in section 4.4: -

Motor controller register value			Feedback on Encoder				
(MSB)	(LSB)	Value	Channel A (Hz)	Index(Hz)	Speed (rpm)	radian/s	rpm/value
0x00	0x64	100	2464	1.23	73.92	7.741	0.07741
0x00	0x78	120	2953	1.48	88.59	9.277	0.07731
0x00	0x8C	140	3457	1.73	103.71	10.860	0.07757
0x00	0xA0	160	3994	2.00	119.82	12.548	0.07842
0x00	0xB4	180	4508	2.25	135.24	14.162	0.07868
0x00	0xC8	200	4990	2.50	149.70	15.677	0.07838
0x00	0xDC	220	5487	2.74	164.61	17.238	0.07835
0x00	0xF0	240	6033	3.02	180.99	18.953	0.07897
0x01	0x04	260	6549	3.27	196.47	20.574	0.07913
0x01	0x18	280	7030	3.52	210.90	22.085	0.07888
0x01	0x2C	300	7510	3.76	225.30	23.593	0.07864
0x01	0x40	320	8072	4.04	242.16	25.359	0.07925
0x01	0x54	340	8590	4.30	257.70	26.986	0.07937
0x01	0x68	360	9074	4.54	272.22	28.507	0.07919
0x01	0x7C	380	9545	4.77	286.35	29.987	0.07891
0x01	0x90	400	10109	5.05	303.27	31.758	0.07940
0x01	0xA4	420	10637	5.32	319.11	33.417	0.07956
0x01	0xB8	440	11118	5.56	333.54	34.928	0.07938
0x02	0x58	600	15199	7.60	455.97	47.749	0.07958
0x02	0xBC	700	17718	8.86	531.54	55.663	0.07952
0x03	0x20	800	20275	10.14	608.25	63.696	0.07962
0x03	0x84	900	22866	11.43	685.98	71.836	0.07982
0x03	0xE8	1000	25384	12.69	761.52	79.746	0.07975
0x05	0xDC	1500	38114	19.06	1143.42	119.739	0.07983
0x07	0xD0	2000	50778	25.39	1523.34	159.524	0.07976

Table 3. Speed registers value Vs speed (2000CPR)



Graph 2. Speed register value Vs Radian/sec (2000CPR)

For this, the relationship between the speed and the register value (that being obtained at section 4.4) is: -

$$register = 12.506 \bullet Speed + 3.3629$$

In other words, Speed to Register value ratio is: -

$$Speed = 0.0799 \bullet register - 0.2689$$

Register value will now change at about every 0.08 rad/s step changes. The "stair steps" effect is much smaller where truncation into the register value is roughly about 4.5°.

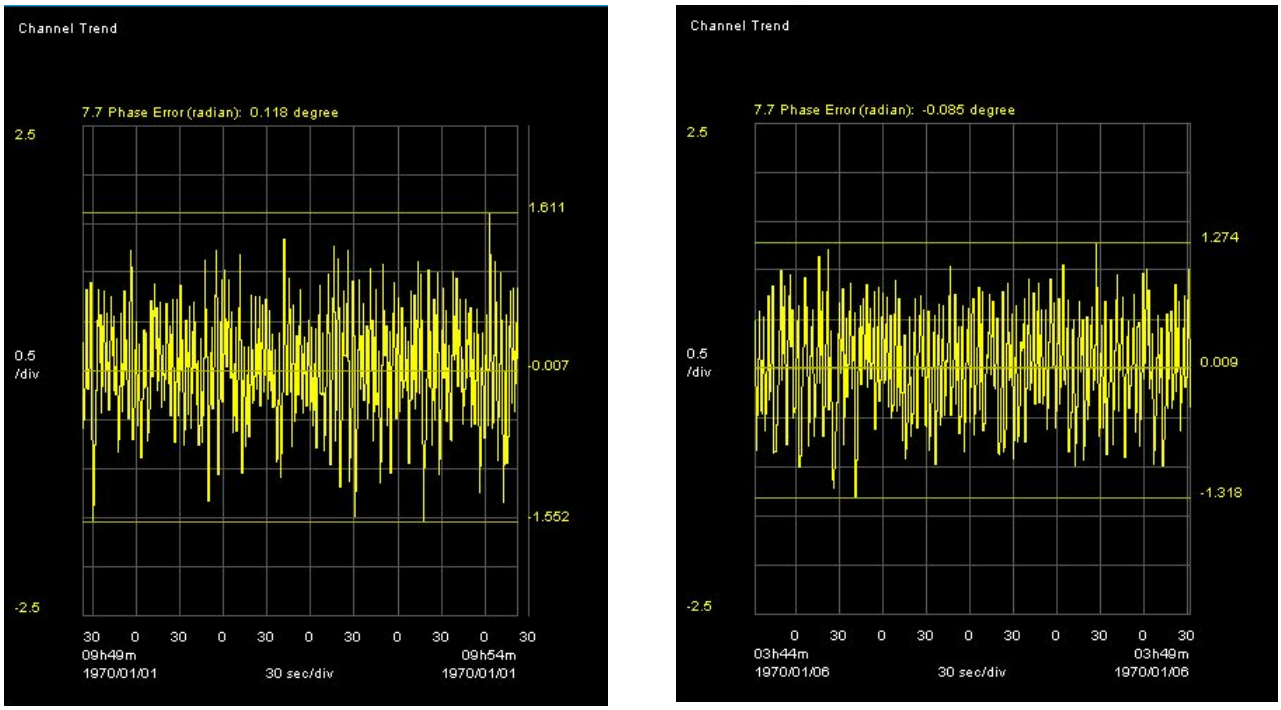
Therefore with this finer resolution of register value over speed, controlling the speed for phase should be more responsive hence reducing the error of the phase angle runs out.

6.4 Second result

The parameter value used is the same as the first experiment.

6.4.1 Phase control Accuracy

Both motor achieve the accuracy of the phase angle of about **±1.5°**.



Motor 1 Motor 2

Figure 15: Phase Control Accuracy at using 2000CPR encoder

6.4.2 Rotation Synchronization result

On the laser measurement plot, we notice a consistent measurement of speed at about 480 rad/s: -

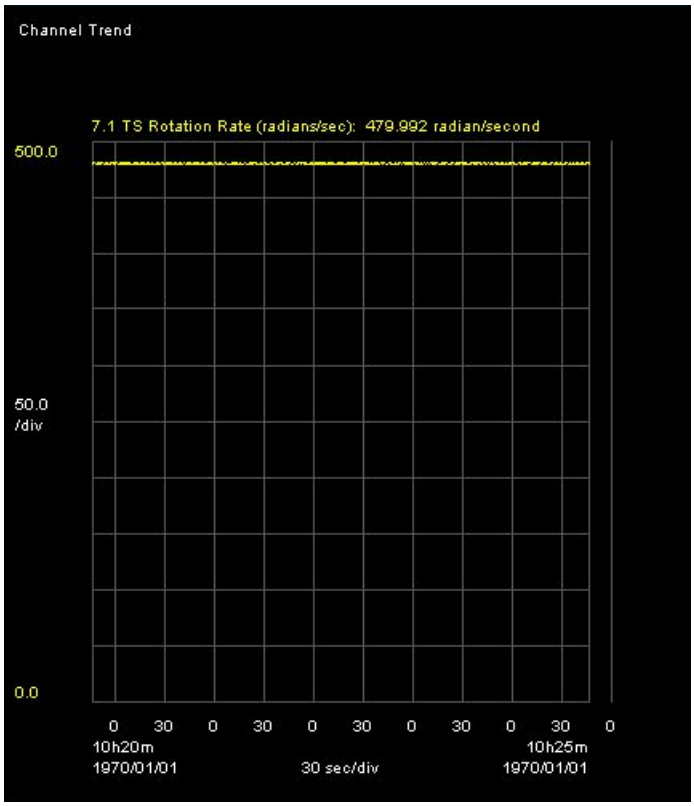


Figure 16: Rotation Synchronization at using 2000CPR encoder

6.4.3 Observation

The second result shows that the rotational synchronization is achieved consistently, where the speed measured at 480 rad/s all time, where both flywheels are lined up slot-to-slot. As the phase control accuracy is well below the opening slot angle, the rotational synchronization is achieved.

6.5 1588 Clock Synchronization effect Test

By using the second experiment setup, this test is the monitor the effect of 1588 Clock synchronization protocol playing its part in the rotational synchronization.

6.5.1 Unplugged Motor 1 node unit from the network

This test is to unplug the Motor 1 1588 node unit from the network, hence stopping it from participating in the 1588 clock synchronization. By monitoring the speed measured by the laser beam, rotational synchronization behavior can be observed.

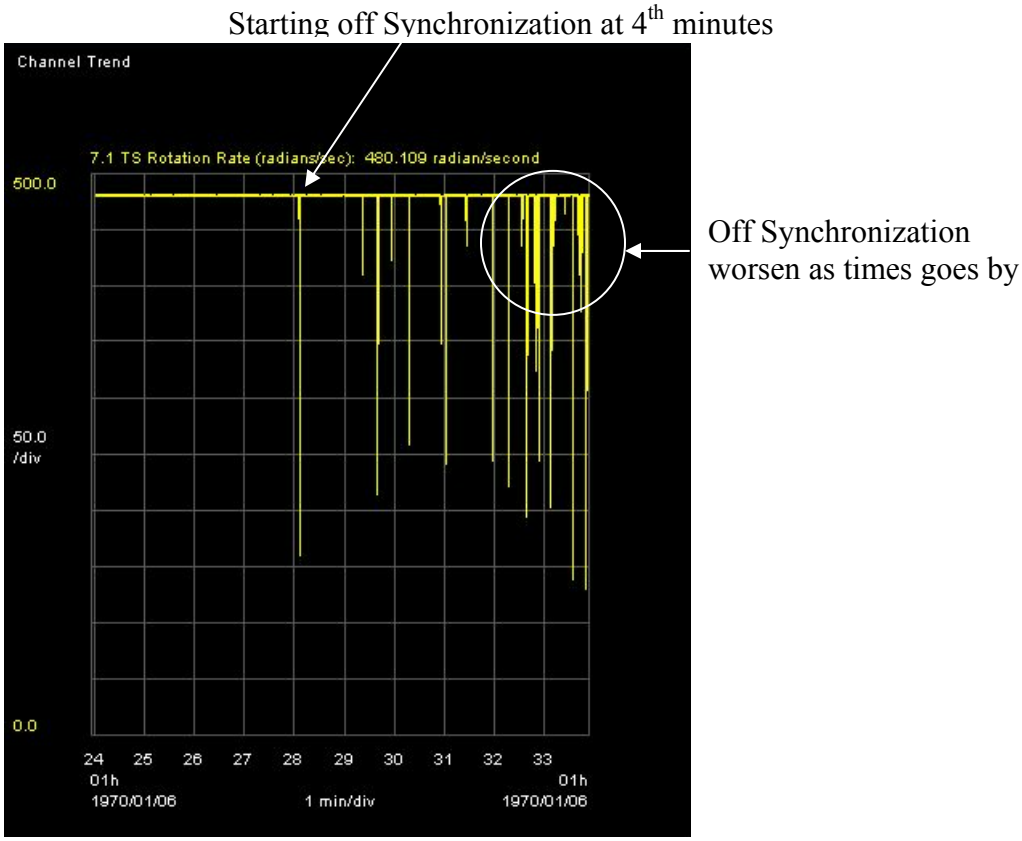


Figure 17: Rotation Synchronization when unplugged Motor 1 node

The rotational synchronization starting the run out starting at the 4th minute from the time the Motor 1 node is unplugged from network. At the plotting graph, the speed measured dropped momentously out from 480 rad/s (synchronized speed) that creates vertical lines downwards shows that the synchronization is out. It got worsens as time passed.
However referring to the node itself, the motor phase angle control is well in place. The run out is due to the absence of 1588 clock synchronization, that the Motor 1 node clock is shifted offset from the master clock.

6.5.2 Plugging Motor 1 node unit to the network

Continue from the previous test, the Motor 1 node is then plugged back to the network hence making it participate back to 1588 clock synchronization.

The result on the laser beam measurement is shown in Figure 18.

From the preceding run out situation due to the absence of 1588 protocol, the rotational synchronization is out as the vertical lines that represent measurement of the speed dropped as the slot to slot is out. As soon as almost 2 seconds after the Motor 1 node plugged back to network, enabling 1588 clock synchronization, the laser beam speed is measured back to 480 rad/s consistently (synchronized speed) as both node times are synchronized.

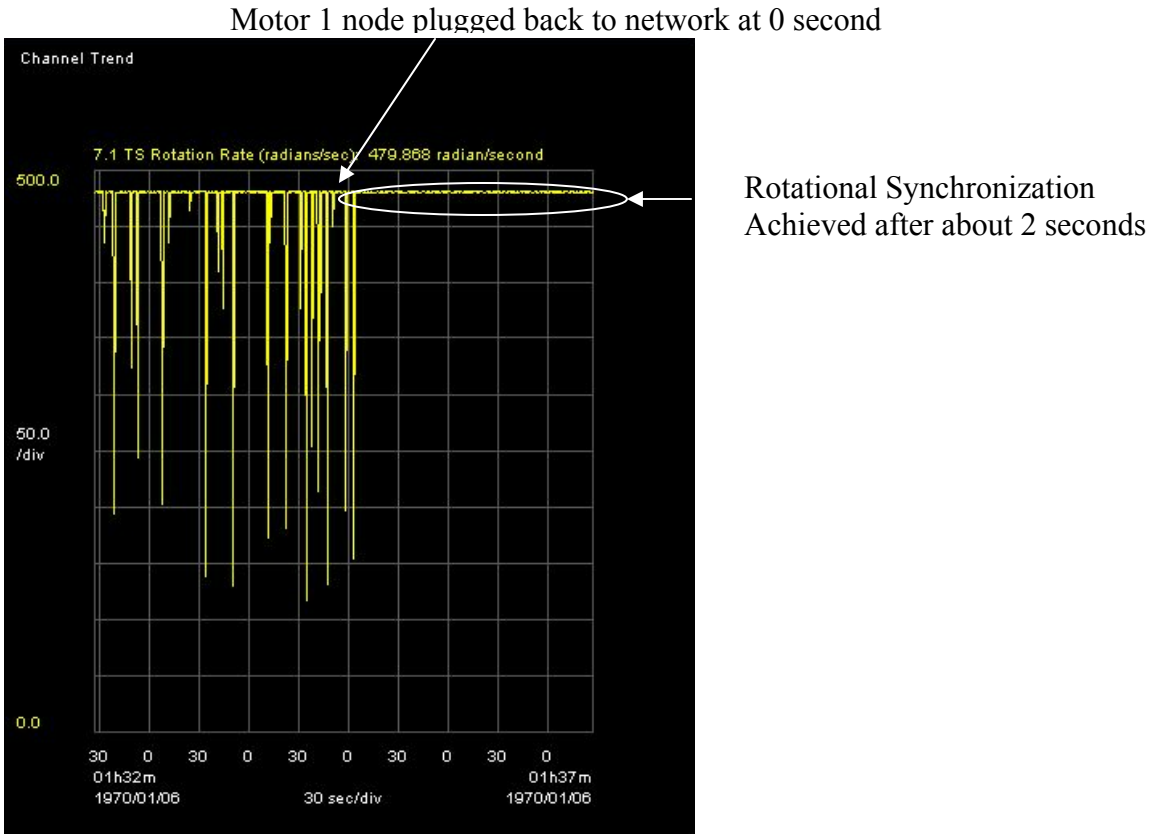


Figure 18: Rotation Synchronization when plugged back Motor 1 node

6.5.3 Rotational Synchronization Drift Limit

From the observation on the 1588 Clock effect test, the motors took some time to be drifted out rotationally, which is the clock drift limit for rotational synchronization. This drift limit can be calculated based on the rotation speed of the motor and the phase control accuracy as below relationship: -

$$DrfitLimit = \frac{PhaseAccuracy}{360^0} \times RotationPeriod$$

As by the current setting of rotation speed of 40 rad/s, the rotation period is ~**157ms**, and the phase angle accuracy is at **±1.5°**, therefore: -

$$DrfitLimit = \frac{1.5}{360^0} \times 0.157\,\text{sec} \approx 654us$$

6.6 Conclusion

In this 1588 unplugged test, it shows that the 1588 clock synchronization is essential in achieving the rotational synchronization. Both motors are running their phase control separately, and even though running with the same parameters, it could not establish a consistent rotational synchronization. With 1588 clock synchronization, the rotation between these motors can be then synchronized at all time.

7.0 Summary & Conclusion

7.1 Summary of paper

This paper has described the methodology of controlling a BLDC motor phase angle base on the time. It controls the motor to be always rotating to a certain mechanical degree. With IEEE1588 protocol, both nodes, which control separate motors, will be in synchronization of their respective time. This means that their time is closely tight together as 1588 protocol synchronizes their local clocks. Therefore, the flywheels attached to the motors thus will be rotating at the same phase, achieving the rotational synchronization.
Without time synchronization, as the phase angle of motor is derived from the absolute time of local time, those motors will not be rotating in the same phase.

7.2 Constraints of current work

In this demonstration, there are TWO main constraints: -

1) Motor phase angle control accuracy due to encoder resolution
2) Rotational synchronization drift limit due to rotation speed

Firstly, Motor phase angle control is being achieved through the oscillation of speed sent to the motor controller board. The motor controller use the encoder feedback as the input for new speed adjustment, thus the encoder resolution is affecting the accuracy of the motor phase angle control. At first, the encoder that built-in on the motor is only 500 count-per-revolution (CPR), giving wide offset of **±12°**. With a second encoder attached to the motor with 2000 CPR, the offset can then be reduced to **±1.5°**. Therefore, the accuracy of the phase angle control is depended on the encoder resolution. A higher CPR value should result in smaller offset.

Secondly, as per the phase angle offset is **±1.5°**, which in time of rotation is about **±654us**. As the synchronization accuracy is within 100ns, therefore after we had stop the 1588 protocol, the system will take quite some time to be off from the rotational synchronization as when the clock that in slave state drift away from the master clock. A faster rotation speed of the motor should be used to further match the accuracy of the 1588 clock synchronization.

tiSSio
ENaBle NetworkinG

IEEE1588 Conference & Plug-Fest Presentation

Rotational Synchronization via IEEE1588

By Keen-Hun, Leong

Agilent Technologies – Motion Control Products Division

tiSSio
ENaBle NetworkinG

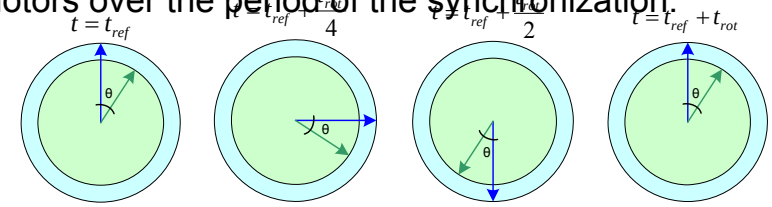
Objective:

- Establish a method to carry out rotational synchronization of two BLDC motors across Ethernet, based on IEEE1588 Clock Synchronization

tiSSio
ENaBle NetworkinG

Introduction:

- Rotational synchronization is defined as 2 or more motors, each rotating at the certain phase angle, mechanically at the encoder index point
- Each motor will have to be well-controlled to achieve the same rotation period, t_{rot} , to the reference of a same starting time, t_{ref} , with a optional offset phase angle, θ , that is being maintained between these motors over the period of the synchronization.



Agilent Technologies
" AGILENT RESTRICTED

Agilent Technologies – Motion Control Products Division

tiSSio
ENaBle NetworkinG

Introduction:

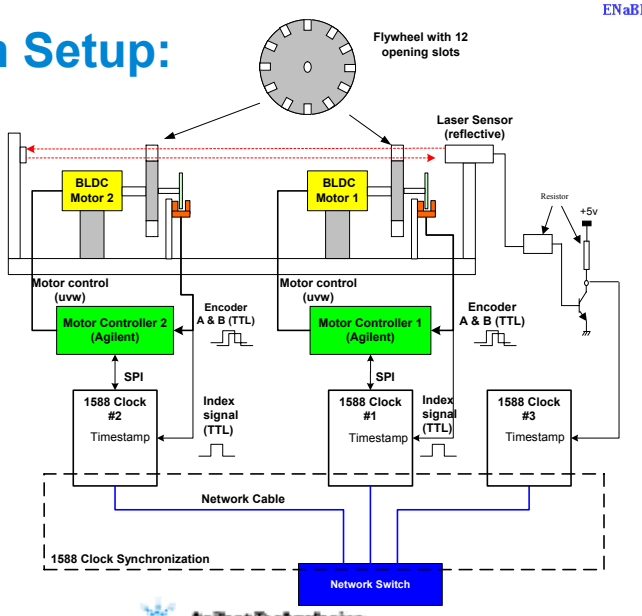
- To achieve Rotational Sync, both motor systems need to be referring to the **same time base** and **well-controlled phase angle**.
- For BLDC motors, it will normally apply speed control and two separate motor control systems will run with a separate time base.
- This demo will utilize the IEEE1588 Clock Synchronization to establish Rotational Synchronization.
- This paper describes the methodology used to derive phase angle of a motor, how 1588 Clock Sync can lead to rotational sync and verification of the measurement.

Agilent Technologies
" AGILENT RESTRICTED

Agilent Technologies – Motion Control Products Division

tiSSio ENaBle NetworkinG

System Setup:



Agilent Technologies

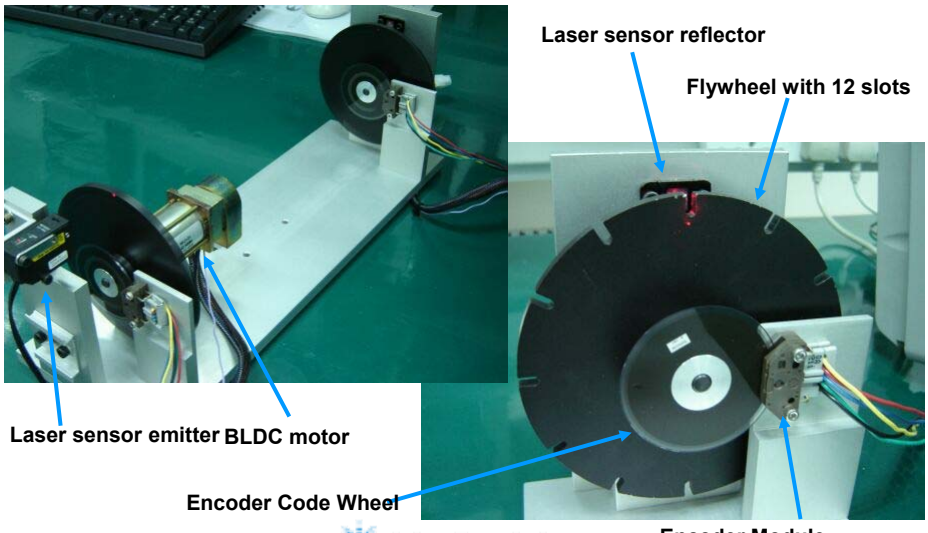
" AGILENT RESTRICTED

Slide 5

Agilent Technologies – Motion Control Products Division

tiSSio ENaBle NetworkinG

System Setup:



Agilent Technologies

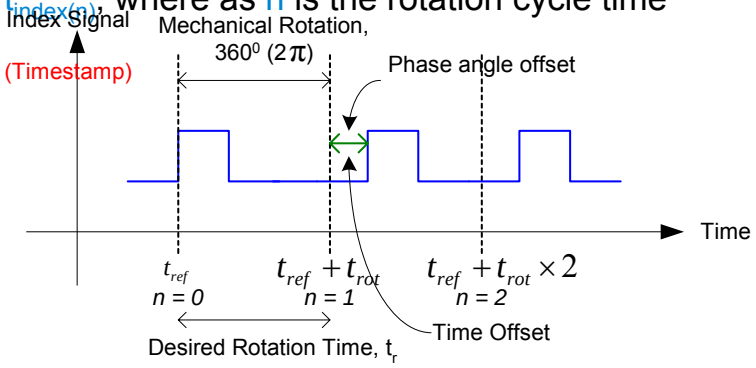
" AGILENT RESTRICTED

Slide 6

tiSSio ENaBle NetworkinG

Rotation Phase Angle:

- Phase angle is the measurement of the offset of rotation index point, in degree, to the desired occurrence time from the reference time, t_{ref} +(t_{rot} x n) or $t_{index(n)}$, where as n is the rotation cycle time

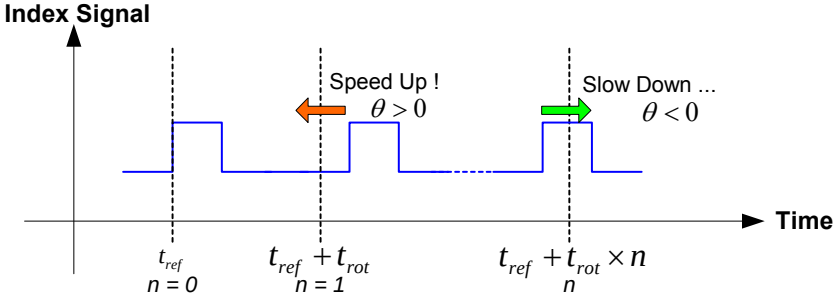


Agilent Technologies
" AGILENT RESTRICTED

tiSSio ENaBle NetworkinG

Phase Angle Control:

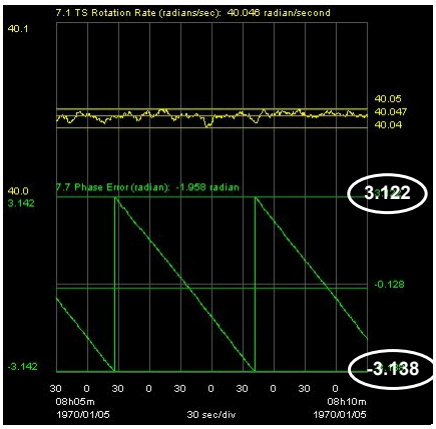
- To adjust the phase angle towards desired value, a simple servo control loop (PI mode) can be applied into controlling the phase angle.



Agilent Technologies
" AGILENT RESTRICTED

tiSSio ENaBle NetworkinG

Speed Control:

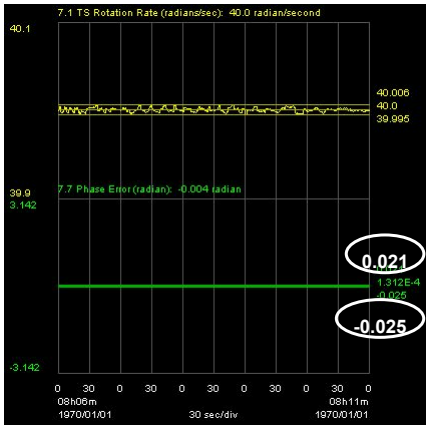


- Motor Control Using Encoder signal A & B
- Speed control accuracy is ±0.01 rad/s (0.095rpm)
- Phase angle is shifting from 3.122 rad to -3.138 rad (~180° to – 180°)

Agilent Technologies
" AGILENT RESTRICTED

tiSSio ENaBle NetworkinG

Speed Control + Phase Angle Control:



- Phase control using Encoder Index signal I in the respective 1588 clock
- Both speed and phase angle are controlled
- Speed control accuracy is ±0.006 rad/s (0.057rpm)
- Phase Angle control accuracy is ±0.025 rad (1.432 degree)

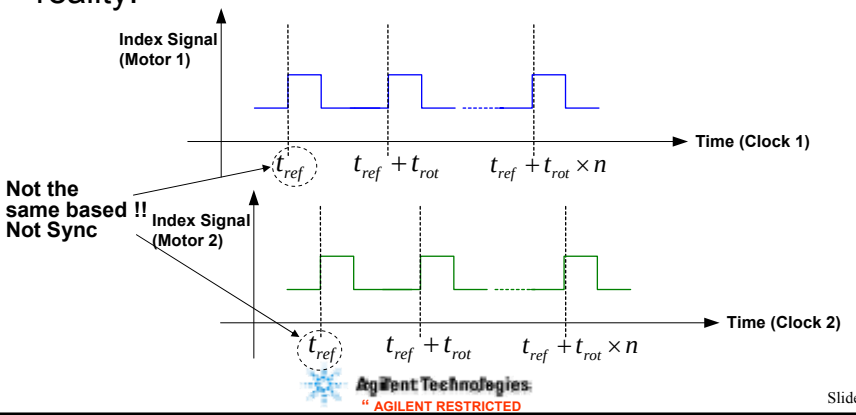
Agilent Technologies
" AGILENT RESTRICTED

Agilent Technologies – Motion Control Products Division

tiSSio ENaBle NetworkinG

Without 1588 Clock Syncronization

- As the phase is derived from a reference time, if both clocks are not synchronized, their mechanical rotation will also not be synchronized, even though both refer to the "same" value of time, which actually is not, in reality.



Agilent Technologies

" AGILENT RESTRICTED

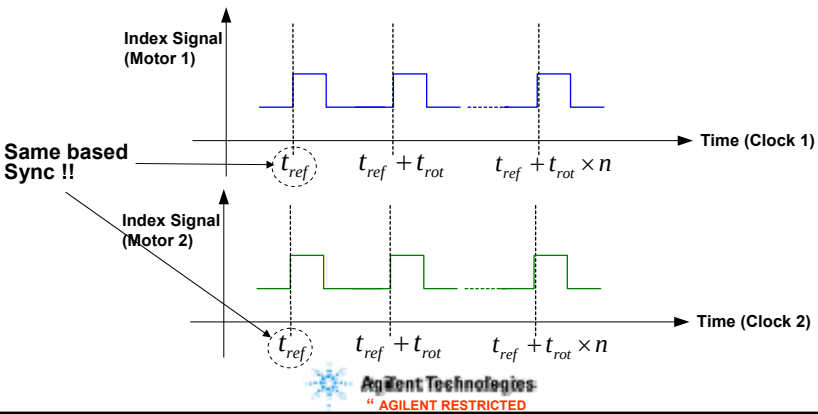
Slide 11

Agilent Technologies – Motion Control Products Division

tiSSio ENaBle NetworkinG

With 1588 Clock :

- With 1588, the time-based is then synchronized, enabling the rotation of both motors to be synchronized.



Agilent Technologies

" AGILENT RESTRICTED

Slide 12

tiSSio ENaBle NetworkinG

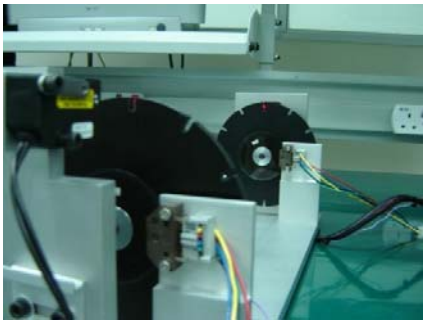
Measurement Method of Rotational

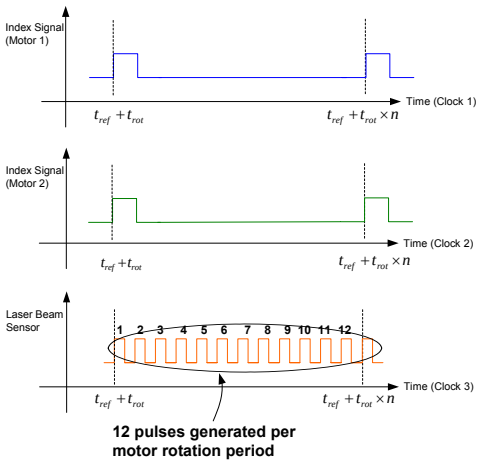
- Method of measurement is by using the laser beam sensor.
- Each motor is attached with a 12-slot flywheel, each slot is a 3^0 opening.
- When both motors are fully synchronized, flywheels are consistently **lined up slot-to-slot** enabling the laser beam to pass through 12 times per rotation.
- The sensor output is connected to another 1588 Clock to measure the speed of synchronization
- Speed measured by the laser beam will be 12 times the speed of the flywheel. Therefore the ability to measure at (**12 x Rotation Speed**) at all times indicates that both flywheels are rotational-

Agilent Technologies
" AGILENT RESTRICTED

tiSSio ENaBle NetworkinG

Measurement of Rotational Sync:



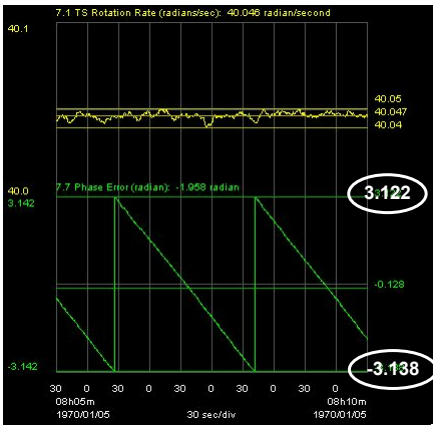


Rotational Sync Out Sync

Agilent Technologies
" AGILENT RESTRICTED

tiSSio ENaBle NetworkinG

Speed Control:

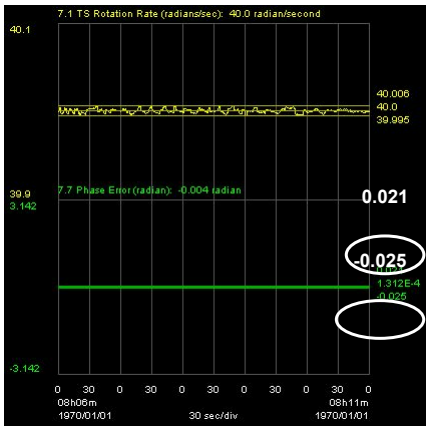


- Motor Control Using Encoder signal A & B
- Speed control accuracy is ±0.01 rad/s (0.095rpm)
- Phase angle is shifting from 3.122 rad to -3.138 rad (~180° to – 180°)

Agilent Technologies
" AGILENT RESTRICTED

tiSSio ENaBle NetworkinG

Speed Control + Phase Angle Control:



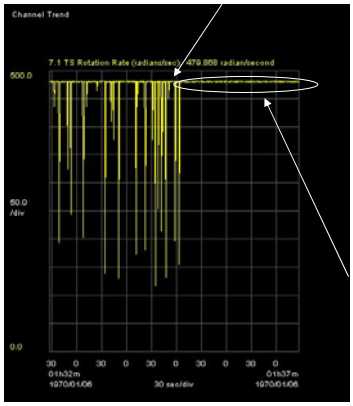
- Phase control using Encoder Index signal I in the respective 1588 clock
- Both speed and phase angle are controlled
- Speed control accuracy is ±0.006 rad/s (0.057rpm)
- Phase Angle control accuracy is ±0.025 rad (1.432 degree)

Agilent Technologies
" AGILENT RESTRICTED

ENaBle NetworkinG

Rotational Sync with 1588 plugged in:

Motor 1 node plugged back to network at 0 second



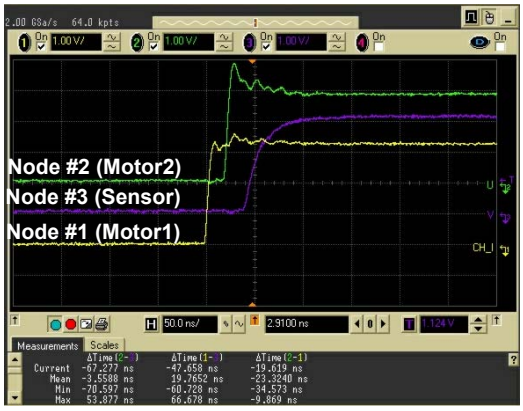
Rotational Synchronization Achieved about 10 seconds later

- At Rotation Off Sync, 1588 node #1 (motor 1) is connected back to the network to enable PTP
- Rotational Sync is then achieved at about 10 seconds after the connection.

Rotation Synchronization when plugged back Motor 1 node

ENaBle NetworkinG

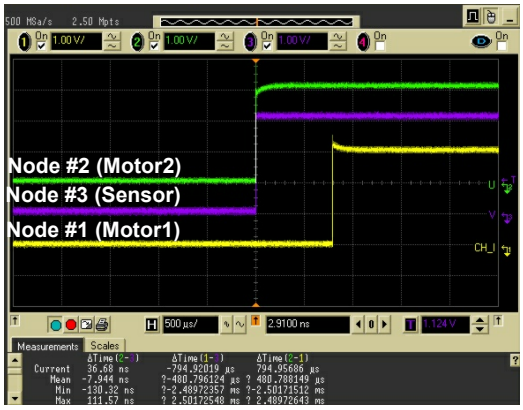
1588 Time Sync ON:



- When time sync is on, the time difference between motor 1 & 2 is <50ns.
- Both motors is in rotational sync.

tiSSio ENaBle NetworkinG

1588 Time Sync OFF:



- Node #1 clock has stop synchronizing to the master Node #2
- When its clock drifts for more than 700us away from Node #2, rotational sync is out.

Agilent Technologies
" AGILENT RESTRICTED

tiSSio ENaBle NetworkinG

Rotation Sync Clock Drift Limit Calculation:

Speed	**40**	rad/s
	381.922	rpm
Rotation Time	0.1571	sec
Phase accuracy	**1.5**	degree
Drift Limit	0.00065	sec
	0.65458	ms
	654.583	**us**
	654583	**ns**

$$DrfitLimit = \frac{PhaseAccuracy}{360^0} \times RotationTime$$
$$= \frac{1.5}{360^0} \times 0.157\,\text{sec} \approx 654us$$

- The clock drift limit for the rotation sync to be out depends on the **rotation speed** & also the **phase accuracy**.
- As the opening of the slot is 3^0, therefore the tolerable phase offset is $\pm 1.5^0$.
- In this demo, the Clock's drifting limit absolute value is ±~650us for the rotation sync to be out.

Agilent Technologies
" AGILENT RESTRICTED

tiSSio ENaBle NetworkinG

Constraints of Applying 1588:

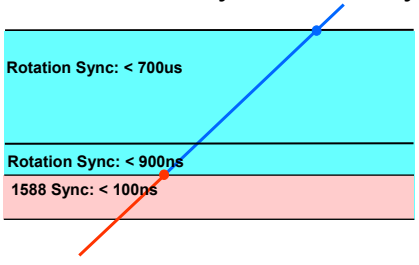
- Motor phase angle control accuracy due to encoder resolution.
  - Motor controller use encoder signals for speed adjustment. By using 500 CPR (Count-per-revolution), phase accuracy is only achieved at ±**12°** but with 2000 CPR, is reduced to ±**1.5°**.
  - Higher encoder resolution will improve phase angle control accuracy.
- Rotation Sync drift limit due to motor rotation speed
  - Rotation speed will determine the drift limit anticipated for the rotation sync to be off. At phase angle accuracy of ±**1.5°** and speed **40** rad/s, which in terms of time is ~ ±**654us**, which is much bigger than 1588 sync accuracy
  - At higher speed the drift limit will be closer to 1588 sync time

Agilent Technologies
AGILENT RESTRICTED

tiSSio ENaBle NetworkinG

Illustration of Constraints:

- With higher rotational speed & better phase angle accuracy, it can lower the drift limit (<700us) to nanosecond range (<900ns), then it can approach the 1588 clock sync accuracy (<100ns)



Speed	**200**	rad/s
	1,909.612	rpm
Rotation Time	0.03142	sec
Phase offset	**0.01**	degree
Drift Limit	8.728E-07	sec
	0.0008728	ms
	0.8727778	**us**
	872.77778	**ns**

Agilent Technologies
AGILENT RESTRICTED

Summary & Conclusion:

- With IEEE1588 protocol, both nodes, which control separate motors, will be in synchronization to their respective clock. Therefore, when both motors are rotating at the same phase angle, rotational synchronization is achieved.
- Without IEEE1588 protocol, those motors will not be rotating in the same phase

Acknowledgement:

Agilent Lab:

Jeff Burch – Mentoring & Guidance

Bruce Hamilton, John Eidson, Dieter Vook, Jay Warrior – Guidance & Hospitality

Agilent Malaysia:

Seong-Keat Lee, Swee-Hing Young - Contribution

tiSSio
ENaBle NetworkinG

Question/Feedback ?

Agilent Technologies
AGILENT RESTRICTED

Application of IEEE 1588 to Synchronize Multiple Robot Controllers

Michael Gerstenberger, KUKA Robotics Corporation

Clemens Gmeiner, KUKA Controls GmbH

Stefan Müller, KUKA Controls GmbH

Abstract

There are many industrial robot applications that require synchronization of multiple robot controllers. Examples include load sharing, fixtureless parts mating, and process relative motion such as in multi-robot arc welding applications. Before IEEE 1588, synchronization was typically achieved with custom solutions involving dedicated hardware and additional intercontroller cabling. For robot controllers that already share data via Ethernet, IEEE 1588 is a natural fit to synchronize the controllers. This paper describes our experience implementing IEEE 1588 to replace our earlier custom clock synchronization technique.

The robot controller software runs in the VxWin environment. VxWin enables simultaneous execution of both Windows XP embedded and the VxWorks real-time operating system without compromising the real-time characteristics of VxWorks. Both sides of the VxWin environment can have Ethernet interfaces; the IEEE 1588 protocol is supported on the VxWorks Ethernet port. Both the system clock used by VxWorks and the PC clock used by Windows XP embedded are synchronized to the master clock.

Implementation of IEEE 1588 reduced clock jitter as well as the time needed to reach a synchronized state. It also simplified the process of recovering from the loss of the master clock. Additionally, it enables the controller to synchronize with other devices that support IEEE 1588. With the implementation done entirely in software, we were able to eliminate hardware components inside the controllers and cabling between the controllers that were dedicated to clock synchronization.

Introduction

This paper describes KUKA's experience applying IEEE 1588 to synchronize multiple robot controllers. KUKA's robot controller software runs on the VxWin platform, which combines Windows XP embedded and VxWorks, enabling both to run simultaneously on commercial PC hardware (see Figure 4 and Figure 5). IEEE 1588 is integrated in the real-time software that runs on VxWorks.

Why synchronize robot controllers?

A variety of industrial robot applications can be implemented with synchronized robot controllers. Figure 1 shows a combination of two such examples. In this picture, the two orange robots at left and in the background are cooperating to lift the minivan body. This application is referred to as load sharing. Load sharing can be useful to eliminate the need for specialized fixtures, to lift items that exceed the payload capability of a single robot, or to handle items that are too awkward for a single robot to lift.

When the two robots on the right side of the picture perform an operation on the moving workpiece, they are said to be performing a process relative motion. One example where process relative motions are useful is performing a processing step (for example tack welding) while a part is being transferred to improve cycle time. Another example is using the workpiece carrying robot as a positioner, keeping the workpiece in a particular orientation needed by the application. This can be needed in applications such as arc welding, where it may be important to keep the site horizontal where welding is being performed.

A third example (not shown in the figure) is fixtureless parts mating, where two robots position mating workpieces together, while a third robot processes the mated parts, for example welding them together.

Finally, it may be useful to teach individual robot motions separately, but to program the motions to begin and end simultaneously. In this case, some of the robots will run at reduced speed. This form of synchronized motion is characterized as time-synchronized. This type of synchronization can be found in certain arc welding applications where the motion of an auxiliary axis is to be coordinated with those of two or more robots.

Figure 1. Load sharing and process relative motion example.

How motions are synchronized

The KUKA robot controller software executes motions by computing positions every 12ms. Positions can be computed in either Cartesian space or joint space. For Cartesian motions, an inverse kinematics transformation is performed to produce joint space positions. A further step computes positions of individual motors, taking into account gear interactions. These desired motor positions are then sent to a servo system for further subinterpolation, the result of which produces commands for the servo loops.

For Cartesian motions, the robot controller computes the motion of the tool center point (TCP), defined by the variable $TOOL relative to a frame of reference $BASE (see Figure 2). If the frame of reference is defined to be relative to the faceplate of a second robot, then the position of that robot must be known by the first robot in order to calculate where to place the TCP. The full equation that must be solved is

$$T6 = LK_ROOT:T6M:LK_OFFSET:P: \$TOOL^{-1}$$

where each term is a homogeneous transform and the : operator represents a homogeneous transform multiplication. In this equation, T6 represents the position of the

dependent robot faceplate relative to its base, and T6M is the position of the remote, independent robot's faceplate relative to its base. LK_ROOT represents the position of the base of the independent robot relative to the base of the dependent robot. LK_OFFSET defines the offset of the frame of reference relative to the faceplate of the independent robot. P is the coordinates of a position (including orientation) in space where the TCP of the dependent robot is to be placed. In a load sharing application, P will be fixed during the load sharing motions, resulting in a static relationship between the faceplates of the two robots. In a process-relative motion application, P will change over time, resulting in a process motion overlaid on the moving frame of reference.

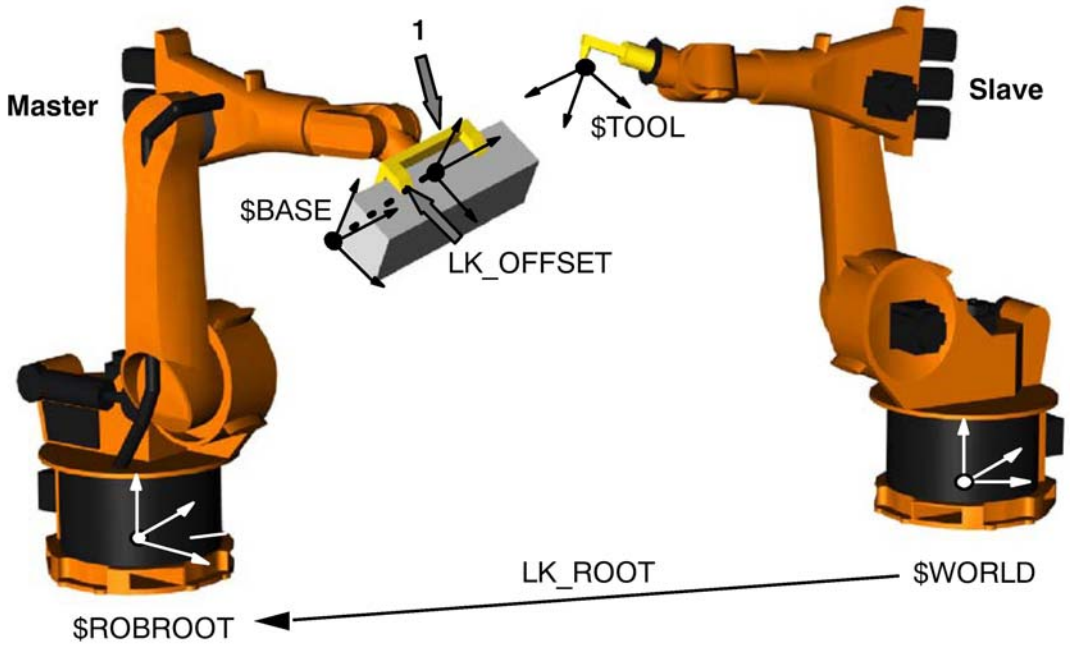


Figure 2. Key frames of reference for geometric coupling.

In order for the dependent robots to complete this calculation, the position of the independent robot must be sent to the other participating robots every interpolation interval (12ms). This is done by sending messages over Ethernet.

The main issue requiring synchronization of the robot controllers is that without synchronization, the clocks on the participating robot controllers will drift apart over time so that their interpolation intervals will no longer line up as in Figure 3. Depending on the drift rates of the robot controller clocks, the dependent robot will eventually either miss a message from the independent robot, or it will receive two messages for a given interpolation interval. Either situation is a problem. In the first case, the velocity of the planned motion would drop to zero for one interpolation interval. In the second case, the

velocity of the planned motion would double for one interpolation interval. These velocity discontinuities can damage the workpiece and even the gears of the robot.

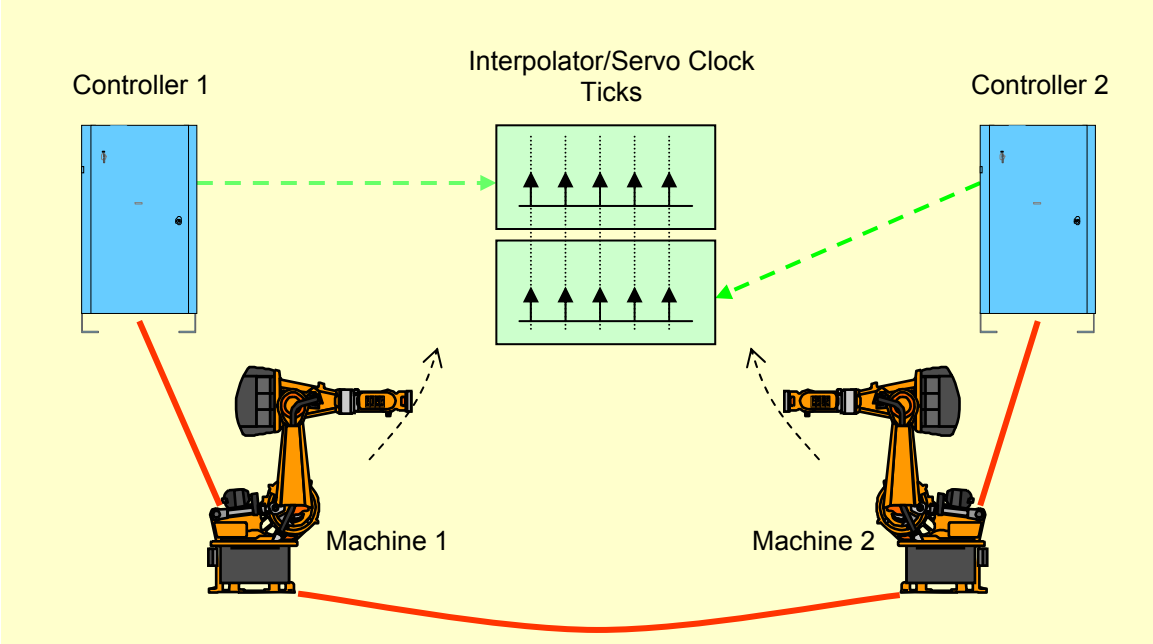


Figure 3. Synchronized interpolation intervals.

In order to smooth out such discontinuities, filters may be used. But these have the drawback of less accurate resultant motion. Certain applications might not be feasible with less accurate motion. For example, inaccuracies in a load sharing application could produce excessive stress on the workpiece, or require tooling with additional compliance to compensate for the inaccurate motion.

Why use IEEE 1588?

KUKA evaluated several techniques for synchronizing the controllers, including distributing the main clock signals of one controller to all the participating controllers. The technique currently in production sends a synchronization pulse from a designated master controller to all other participating controllers. This technique produces satisfactory results, but it requires additional custom circuitry in a controller and additional cabling between the controllers, and it consumes resources that are then not available for use by the robot application.

Since RoboTeam already uses Ethernet to send messages among the participating robot controllers, IEEE 1588 is a natural choice. We are able to eliminate the custom circuitry as well as the additional cabling between the participating controllers. This simplifies synchronization of controllers used for simulation purposes, where the additional circuitry and wiring is impractical. Eliminating the cabling means we no longer have a limitation on the number of participating controllers based on the fanout of the sync pulse driver circuitry.

Another advantage of using IEEE 1588 is that we are able to specify commercial components for delivering the synchronization feature. By relying on an external

grandmaster clock and Ethernet switches with boundary clock functionality, we can provide a more stable, robust source of synchronization. We also can avoid the situation where taking the clock synchronization master controller offline for maintenance could disrupt all the controllers in a workcell.

KUKA implementation of IEEE 1588

KUKA's synchronization requirements are not especially severe. Instead of an absolute accuracy specification, our requirements derive from needing to keep the 12ms intervals lined up. Our previous synchronization technique achieved jitter of the start of the 12ms intervals of +/- 120µs, so that was our target for an IEEE 1588 based solution. This jitter limit needed to be maintained even under heavy network loading and heavy processor loading. In order to keep cost down, we preferred to avoid specialized hardware (e.g. for providing hardware timestamping of IEEE 1588 messages).

Figure 4 shows a block diagram of the KUKA robot controller, with emphasis on the clocks in the system. The software environment is called VxWin, which combines the VxWorks real-time operating system with Windows XP embedded, all running on a single x86 PC. Communication between software running under Windows XP embedded and software running under VxWorks is achieved via a shared memory TCP/IP interface. Figure 5 shows a variation of VxWin that does not require the custom hardware needed to implement a complete robot controller. More information about VxWin can be obtained from http://www.kuka-controls.com/product/vxwin/ or [2].

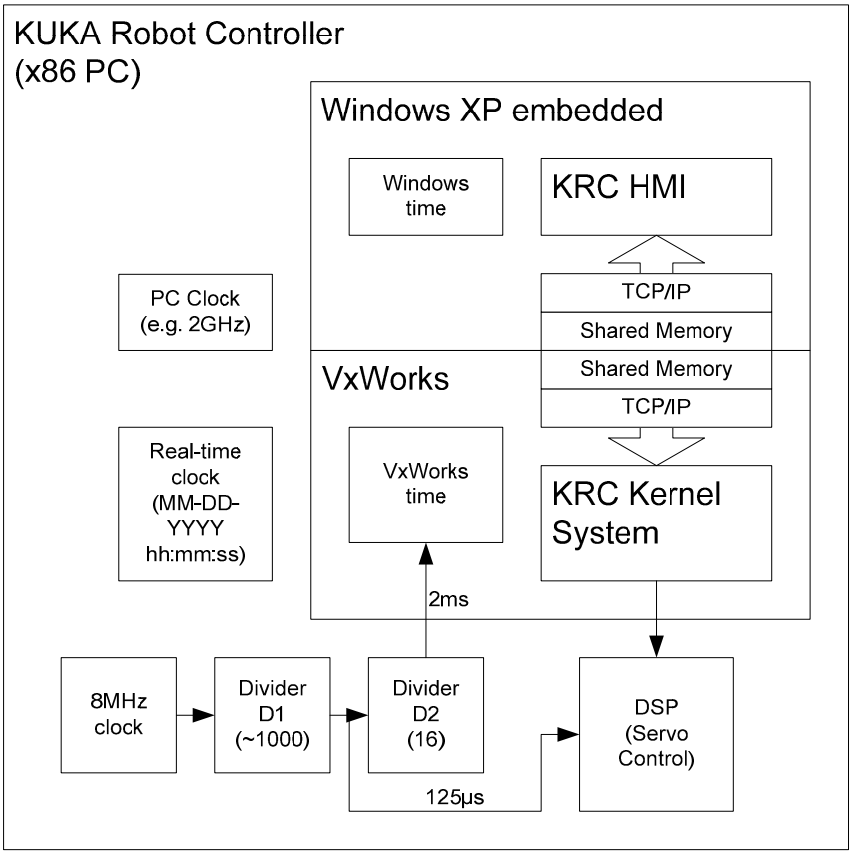


Figure 4. Clocks in the KUKA robot controller.

In each of these systems, VxWorks and Windows XP embedded maintain time information in software, which is seeded at system startup from the real-time clock of the PC. The VxWin environment provides for synchronization of VxWorks time and Windows time. The key to synchronizing PCs is to adjust the rate at which the VxWorks system clock runs. In the case of the KUKA robot controller, this is achieved by adjusting the divider D1, while in VxWin, the divider in the 8253/8254 is adjusted.

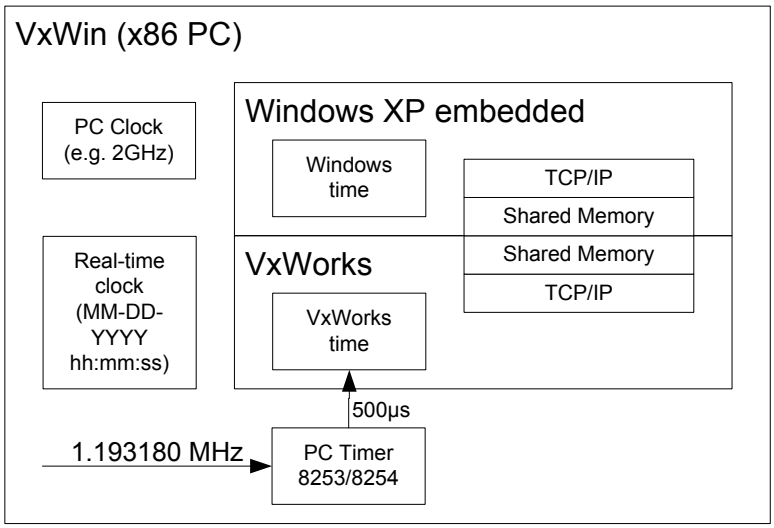


Figure 5. Clocks in the VxWin system.

KUKA's implementation of IEEE 1588 is based on the Hirschmann VxWorks implementation. This was adapted to our Ethernet driver (hooks for timestamp capture and retrieval) and to our clock adjustment algorithm. Our implementation is a software-only solution; that is, it does not require that the Ethernet interface provide for capturing of timestamps in hardware.

In our previous clock synchronization technique, selection of the clock sync master is done manually. This was preserved in our IEEE 1588 implementation by noting that a clock with stratum 255 shall never be the best master clock [1]. We therefore use the default stratum of 255 for all our clock sync slaves, and set the desired master to have a stratum of 4.

Because our primary interest is in lining up the 12ms intervals, we were able to tune our clock adjustment algorithm for relatively fast startup. At startup, most of the time difference between the grandmaster and the local clock is applied immediately. The portion of this time difference that is a fraction of the 12ms interval is corrected for over a configurable interval. During the initialization phase, we also measure the drift between the local clock and that of the grandmaster. The drift correction is applied repeatedly over a configurable interval, usually the sync interval. Each sync message supplies information about an additional phase correction, which is applied over another configurable interval.

Principal challenges

The principal challenge we faced was the coarse resolution of our clock dividers. In the KRC implementation (Figure 4), although our primary VxWorks system clock runs at 2ms from an 8MHz source, we are constrained to adjusting the divider D1 (nominally 1000), since the 8KHz clock drives our servo system, which must stay synchronized with VxWorks. Given the 100ppm accuracy of our oscillators, this means that the minimum correction we can apply is 10 times the expected error. Alternatively stated, we expect to only apply a correction fewer than one 2ms tick out of ten.

The situation is worse when using the PC timer. With the PC timer, the nominal divider value is 596, so the minimum correction step size is 17 times the expected error.

The coarse resolution of the correction we can make has a big impact on the jitter that we observe. Each 2ms tick in which we decrease the divider by one count causes the clock to race ahead at generally 10 (17) times as fast as what we would really like. Then we return the divider to its nominal value for several ticks in which the clock slows back down.

Another challenge occurred when using the PC timer, where the observed clock drift was much higher than expected, in the range of +/- 500ppm. The nominal observed clock drift for the KRC implementation was +/- 30ppm.

Experimental results

We measured the alignment of our time intervals by outputting a short pulse on the LPT port of the PC when the VxWorks sysClk interrupt occurs. We measured the variability of this pulse (jitter) under a variety of conditions. Each measurement tested VxWin outputting a pulse every 1ms and a KRC clock sync master and slave outputting a pulse every 2ms. The tests used the envelope function of the oscilloscope to capture the worst case jitter. Tests were run for approximately 15 minutes each. Processor loading was achieved by a high priority task blocking other tasks for 4ms. Network loading was achieved by directing ICMP traffic at the port under test.

In each of the following oscilloscope plots, the VxWin slave is shown at the top, the KRC slave is in the middle, and the KRC master is on the bottom. The cursors are placed approximately at +/- 70μs from the center, except for Figure 8 center.

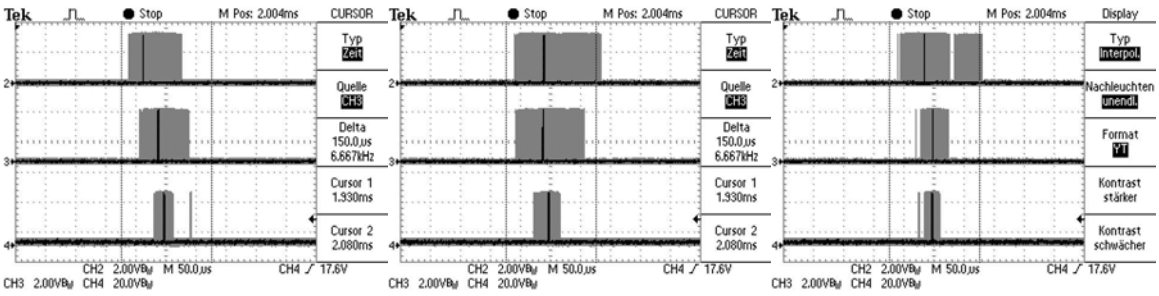


Figure 6. Effect of network blasts directed at left) KRC slave, center) KRC master, right) VxWin slave.

In Figure 6, we compare the effect of network blasts directed at the 3 different nodes. Not surprisingly, network traffic directed at the slave nodes degrades their performance. However, network traffic directed at the master degrades the performance of both slaves. This is most likely due to the network traffic interfering with the quality of the timestamps produced by the master.

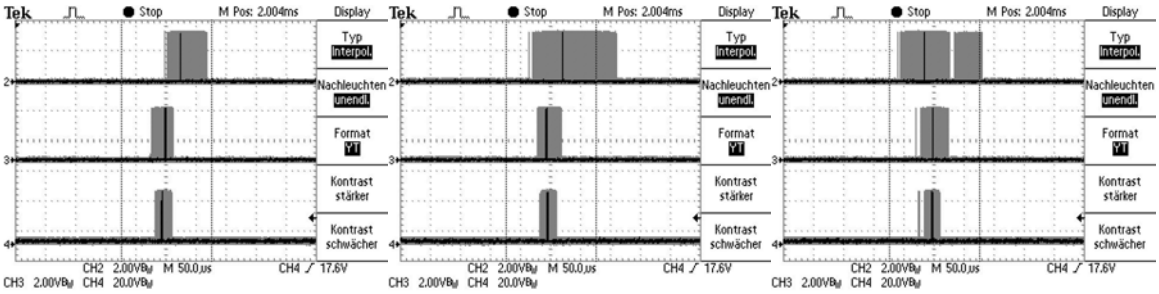


Figure 7. Effect of network load: left) without load, without driver timestamping; center) with load, without driver timestamping; right) with load, with driver timestamping.

In Figure 7, we compare the effects of network loading, with and without driver timestamping. Network load is shown to have the expected effect, which can be mitigated by introducing driver timestamping.

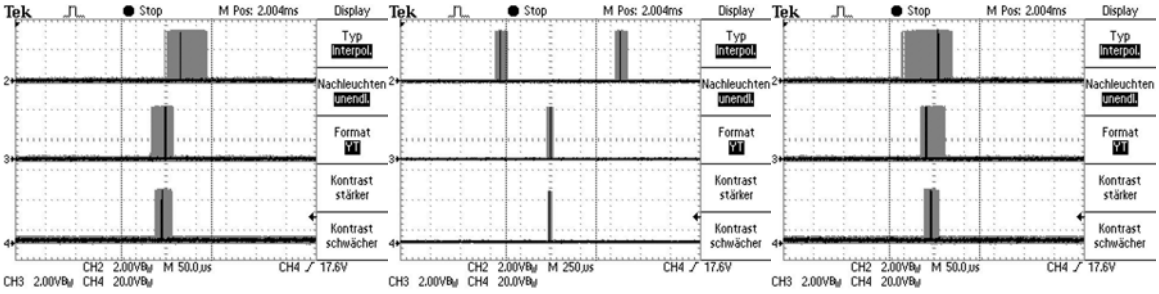


Figure 8. Effect of processor load: left) no load, without driver timestamping; center) with load, without driver timestamping; right) with load, with driver timestamping.

In Figure 8, we compare the effects of processor loading, with and without driver timestamping. Without driver timestamping, processor loading causes a complete breakdown of synchronization; the relative times of the pulses are stable, but the phasing is incorrect. Note that the timescale is changed in Figure 8 center. As expected, driver timestamping is seen to improve the performance.

Conclusions

We were able to meet our target jitter of +/- 120µs. In fact, our implementation of IEEE 1588 produced even better jitter, at +/- 70µs. Meeting our jitter goal enables us to eliminate the dedicated hardware and intercontroller cabling required for distributing a synchronization signal, as well as to free up a COM port for use by robot applications.

Driver timestamping was found to be essential to producing acceptable results in the presence of network and processor loading.

Jitter was found to be extremely sensitive to the clock divider resolution. This was especially challenging when using the 8253/8254 PC timer. Another factor was that the PC timer frequency was not an integer multiple of the VxWorks sysClkRate. Even if the clocks were perfect, this would require continual adjustment of the divisor to achieve the desired sysClkRate. An area of future work will be to implement support for the APIC, which should produce better results.

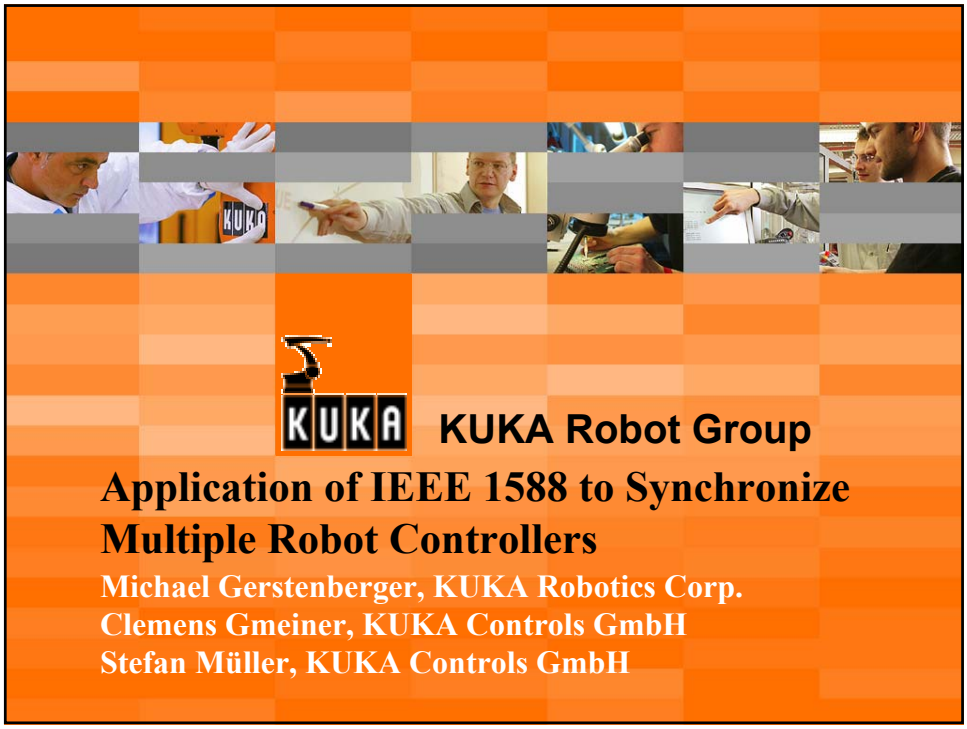
Future development will also target smooth transitions from one grandmaster to another without loss of synchronization.

References

[1] IEEE Std1588-2002, "Standard for a precision clock synchronization protocol for networked measurement and control systems"

[2] John S. Patchin, "Running Windows and Real-Time on the Same Computer", *Embedded Design Focus---Automotive*, September 2005, pp. 13-18.

[3] "KUKA.CR RoboTeam Overview", 2004, KUKA Robotics GmbH



KUKA Robot Group

Application of IEEE 1588 to Synchronize Multiple Robot Controllers

Michael Gerstenberger, KUKA Robotics Corp.
Clemens Gmeiner, KUKA Controls GmbH
Stefan Müller, KUKA Controls GmbH

Why synchronize robot controllers?

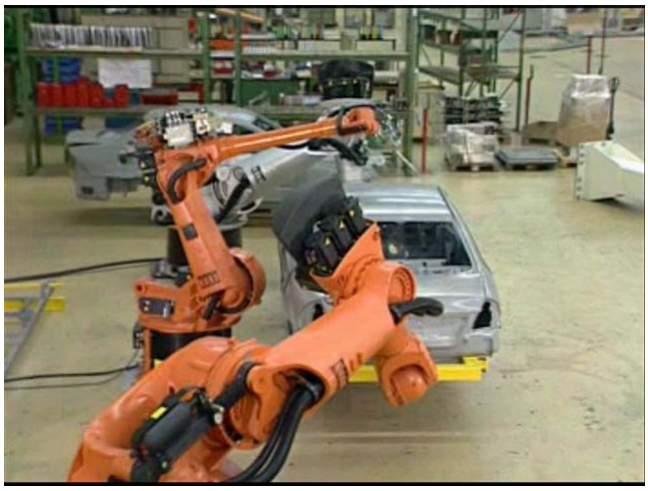
- KUKA RoboTeam Motion Cooperation
  - Geometric coupling
    - Load sharing
    - Process relative motion
  - Time-synchronized motions
- Correlation of events occurring on different controllers
  - Useful for diagnostics

© 2005 KUKA Robotics Corp.

RoboTeam in action: Load sharing

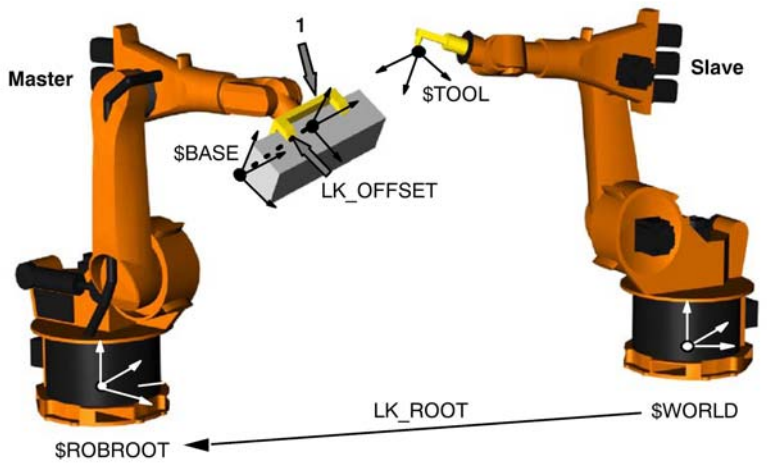
 © 2005 KUKA Robotics Corp.

RoboTeam in action: Process relative motion



 © 2005 KUKA Robotics Corp.

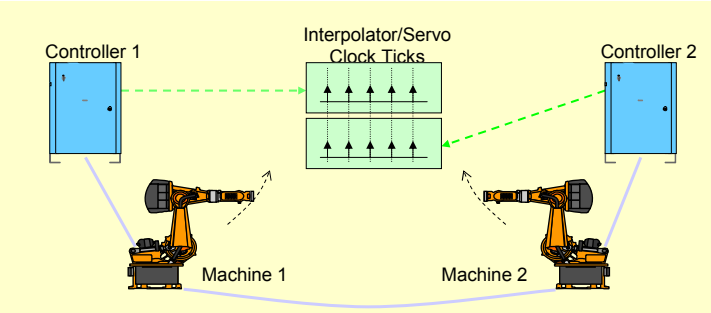
How motions are coordinated: positions



 © 2005 KUKA Robotics Corp.

How motions are synchronized: clocks

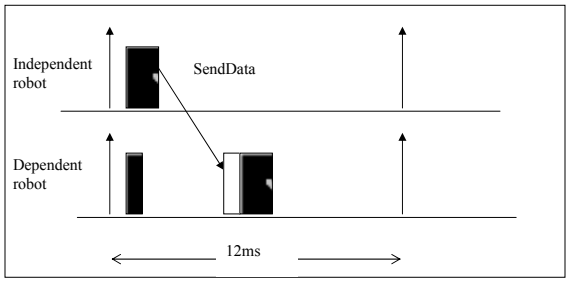
- V = Δd / Δt
- 12ms @ 2m/s = 24mm
- Missing tick represents 12ms @ 0 speed
- Extra tick represents 12ms @ double speed



 © 2005 KUKA Robotics Corp.

How motions are synchronized: messages

- Each controller computes where it wants to be every 12ms
- Dependent robots must wait for position information via Ethernet messages from robot to which they are linked
- Multiple levels of dependency are permitted



 © 2005 KUKA Robotics Corp.

Why use IEEE 1588?

- Ethernet already being used for messages among cooperating robot controllers
- Eliminates custom components needed for previous synchronization technique
- Frees up system resources dedicated to previous synchronization technique
- Fanout of synchronization signal no longer an issue
- Use of published standard improves interoperability / selection of external components

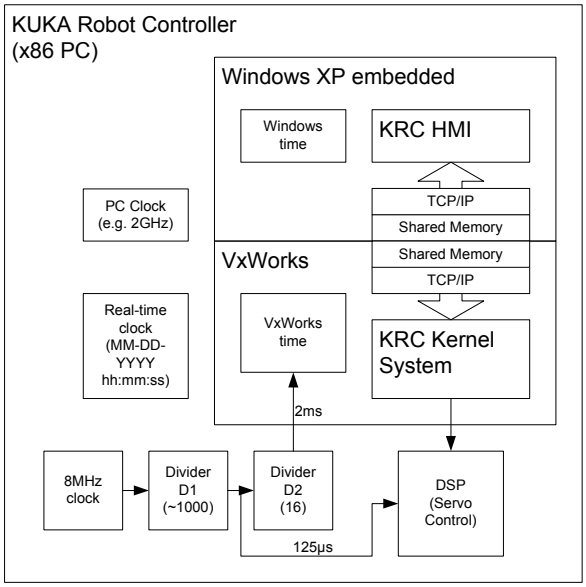
 © 2005 KUKA Robotics Corp.

Synchronization Requirements

- Goal: synchronize 12ms interpolation intervals
- Knowledge of absolute time not essential
- Discontinuous time adjustment at startup acceptable
- No more than +/- 120µs jitter between start of each controller's 12ms interval
- Jitter specification must be met under high processor loading and high network loading

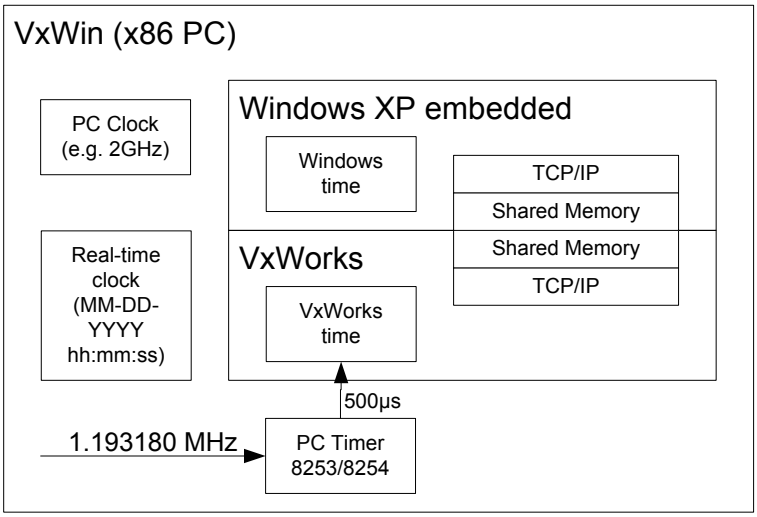
© 2005 KUKA Robotics Corp.

Clocks in the KUKA Robot Controller



© 2005 KUKA Robotics Corp.

Clocks in VxWin



 © 2005 KUKA Robotics Corp.

KUKA implementation of IEEE 1588

- Adaptation of Hirschmann implementation to the KRC & VxWin platforms
- Software-only solution (no hardware timestamping)
- Hooks added:
  - Record timestamp on message receive (in ISR)
  - If received packet = Sync or DelayReq, retrieve timestamp
  - If sent packet = Sync or DelayReq, record timestamp (used later in Followup packet)
- Clock adjustment algorithm
  - Measure drift at startup
  - At startup, time difference that is a multiple of 12ms is applied instantaneously; remainder is applied over a configurable interval
  - Drift correction applied every 2ms over configurable interval
  - Phase correction applied every 2ms over configurable interval

 © 2005 KUKA Robotics Corp.

Principal challenges

- Clock divider resolution
  - 100ppm clocks, main clock divider of 1000 → expected drift-corrected divider of 999.9 to 1000.1 for perfect grandmaster, or 999.8 to 1000.2 for a 100ppm master with maximum error
  - Alternatively, 1 count on clock divider = 10 x clock accuracy
  - Worse for VxWin, with nominal divider of 500µs x 1.19318MHz = 596; 1 count on clock divider = 17 x clock accuracy
- Clock drift
  - +/- 30ppm (observed) for KRC
  - +/- ~500ppm (observed) for VxWin

 © 2005 KUKA Robotics Corp.

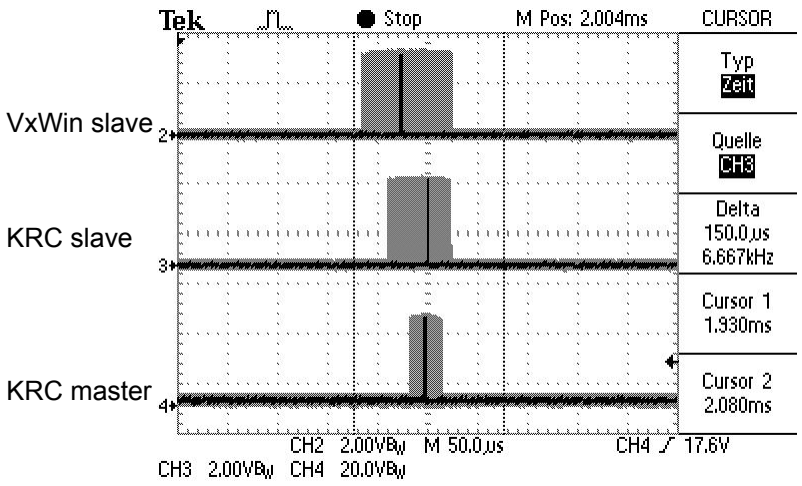
Experiments

- Performance of KRC vs. VxWin
- Performance under heavy network load
- Performance under heavy processor load
- Performance with and without driver timestamping

- Test conditions:
  - VxWin @ 1ms, KRC @ 2ms
  - “Heavy processor load” is task blocking for 4ms
  - “Network blast” is ICMP traffic directed at Ethernet port

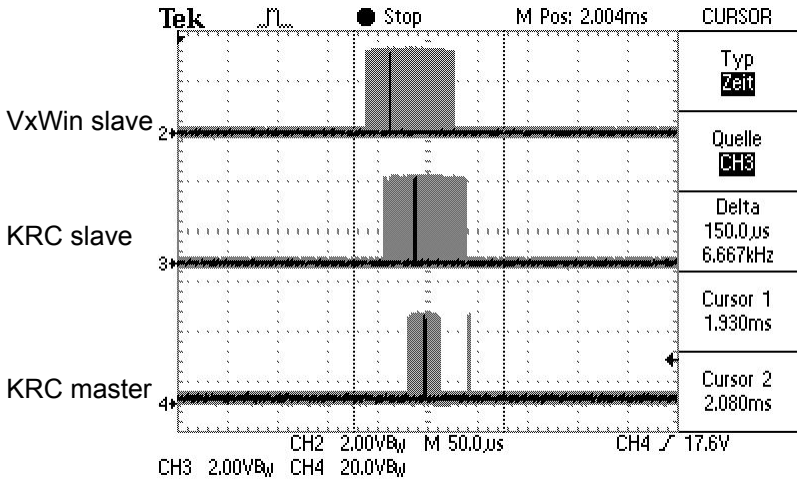
 © 2005 KUKA Robotics Corp.

Basic jitter result



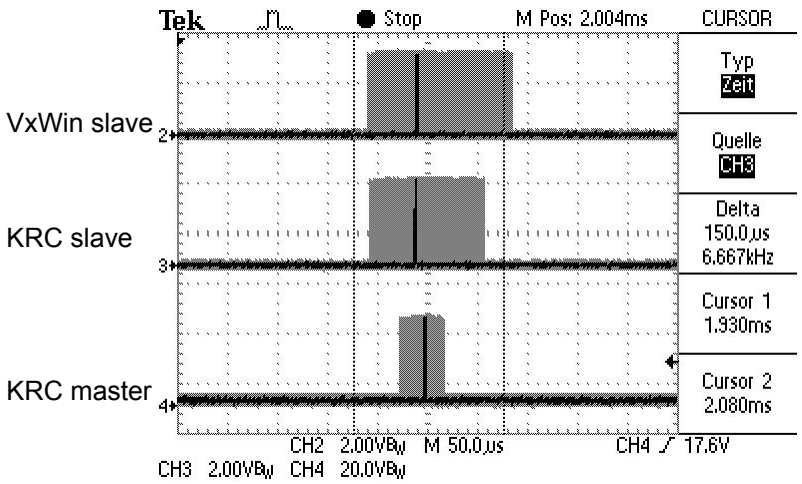
© 2005 KUKA Robotics Corp.

Network blasts on KRC slave



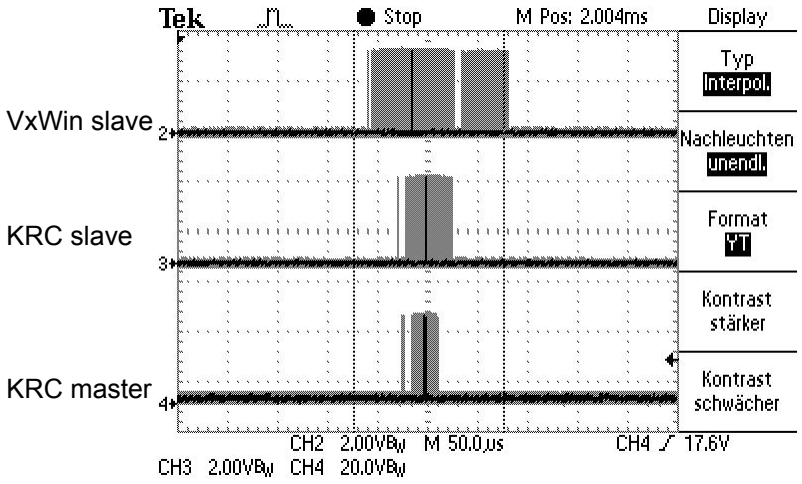
© 2005 KUKA Robotics Corp.

Network blasts on KRC master



© 2005 KUKA Robotics Corp.

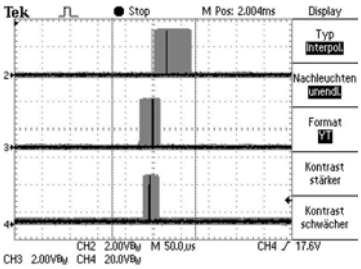
Network blasts on VxWin slave



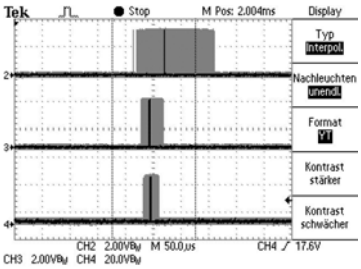
© 2005 KUKA Robotics Corp.

Effect of network load, VxWin without driver timestamping

Without network load



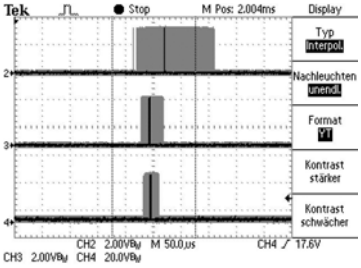
With network load



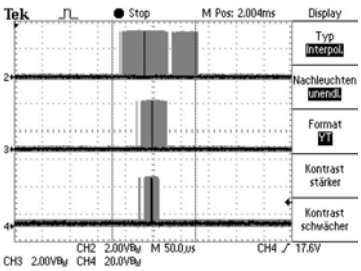
© 2005 KUKA Robotics Corp.

Effect of driver timestamping, VxWin with network load

Without driver timestamping



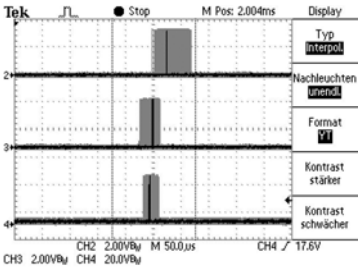
With driver timestamping



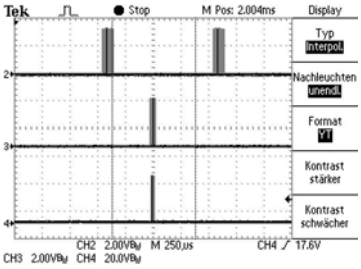
© 2005 KUKA Robotics Corp.

Effect of processor load, VxWin without driver timestamping

Without processor load



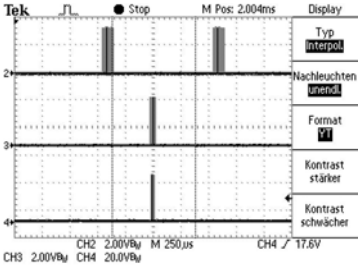
With processor load



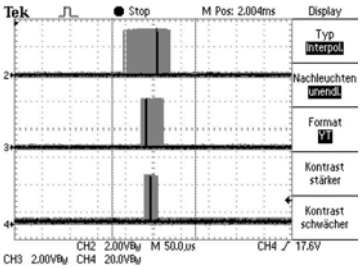
© 2005 KUKA Robotics Corp.

Effect of driver timestamping, VxWin with processor load

Without driver timestamping



With driver timestamping



© 2005 KUKA Robotics Corp.

Conclusions

- Target of +/- 120µs jitter met; achieved +/- 70µs
- Driver timestamping improves performance
- Jitter is strongly influenced by clock divider resolution
- Dedicated synchronization hardware eliminated/freed up

- Next steps:
  - Improve ability to handle change of grandmaster
  - Implement support for APIC

© 2005 KUKA Robotics Corp.

Migration towards IEEE1588 time synchronization over gigabit Ethernet

Sven Nylund and Øyvind Holmeide

***Abstract* — An IEEE1588 Grand Master and/or Transparency implementation on a gigabit Ethernet (1000BASE) switch or router is a challenge due to the high speed GMII interface between the gigabit switch core and the PHY chip. A high speed and expensive FPGA can be used, but such an implementation should preferably be based on a 1000BASE switch core with IEEE1588 support at PHY level. However, 1000BASE L2/L3 chipset are unfortunately not yet commercially available with IEEE1588 support.**

An intermediate solution based on a 10/100BASE switch/router platform with IEEE1588 support and off-the-shelf 1000BASE switches/routers is therefore proposed in this paper. This solution utilizes that most non-trunk IEEE1588 Slave ports are based on 10/100Mbps speed. VLAN techniques and extra 100BASE trunk links are also used in order to remove any time synchronization degradation through the 1000BASE switches/routers.

***Index Terms* — IEEE1588 over gigabit Ethernet.**

I. INTRODUCTION

The time synchronization accuracy will be degraded if the time synchronization packets are sent through switches and routers without time synchronization properties (e.g. IEEE1588 Transparency). The degradation depends on several factors, where the number of network hops between the IEEE1588 Grand Master and the IEEE1588 Slaves, network load and drop link speed are the most important.
Combining 10/100BASE switches and routers with IEEE1588 support with 1000BASE switches without such properties can be done in order to avoid any time synchronization degradation through off-the-shelf 1000BASE switches and routers.
This requires extra 100BASE trunk links in addition to the 1000BASE trunk links. The IEEE1588 Slave end nodes should preferably be based on 10/100BASE (which is often the case).
Adding extra trunk links will, however, create network loop(s) with broadcast storm as a result unless the loop(s) are handled by using:

1. Layer 2 network redundancy protocol such as Spanning Tree Protocol (STP/RSTP) or similar proprietary protocols (e.g. FRNT from Westermo OnTime).
2. Layer 3 network redundancy protocol such as OSPF and VRRP or similar.
3. VLAN and multicast filtering techniques.

These three alternatives and their corresponding impact on the IEEE1588 time synchronization implementation are discussed below.

II. LAYER 2 NETWORK REDUNDANCY PROTOCOL

Layer 2 network redundancy protocols are based on creating a virtual spanning tree from a network topology, where loop(s) are present. This means that one or more ports will be in "BLOCKING" mode. These ports might be changed to "FORWARDING" mode in case of a topology change. Which port in a loop that will be configured in "BLOCKING" mode depends on user configuration and type of layer 2 network redundancy protocol that is used.
In general no packets except for Layer 2 network redundancy control packets will be sent/received on a port in "BLOCKING" mode. We need to change this if we want to send time synchronization packets through a port in "BLOCKING" mode and other packets on the 1000BASE link as shown in Figure 1.

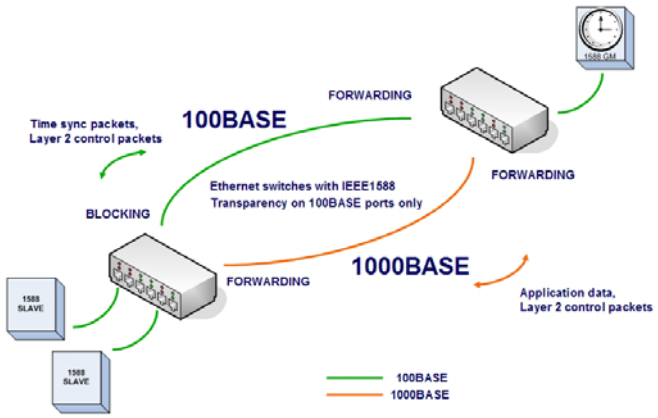


Figure 1, Ring topology handled by a Layer 2 network redundancy protocols; the standby link (100BASE) is used for IEEE1588 time synchronization

A. STP/RSTP

Spanning Tree Protocol (STP) and Rapid Spanning Tree Protocol (RSTP) according to IEEE802.1D and IEEE802.1w respectively are the most used open layer 2 network redundancy protocols.
The trunk port speed will in case STP/RSTP are used, have impact on whether the port shall be configured in "BLOCKING" mode or not. I.e. the STP port weight is higher for a 100BASE port than for a 1000BASE port.

Sven Nylund is a Senior Designer at Westermo OnTime AS.
Sven can be contacted at sven@ontimenet.com.

Øyvind Holmeide is the Managing Director of Westermo OnTime AS.
Øyvind can be contacted at oeyvind@ontimenet.com.

A dedicated multicast MAC address is used for STP/RTSP packets, 0x01 0x80 0xC2 0x00 0x00 0x01, and only packets with this destination multicast address can be sent/received on a port in “BLOCKING” mode. IEEE1588 packets must also use this multicast address in order for this concept to work. This is, however, not good from an IEEE1588 standardization point of view.
Various STP/RSTP parameters can be set in order to influence the STP/RSTP port settings (“FORWARDING” or “BLOCKING” mode) in order to make sure that time synchronization packets are sent on given ports (i.e. 100BASE and not 1000BASE ports). Predicting possible spanning trees based on various topology changes can be done for simple network topologies such as a simple ring topology, while it can hard to predict which port that will be in “BLOCKING” mode if the topology is complex.

B. *FRNT*

The Fast Re-configuration of Network Topology (FRNT) protocol of Westermo OnTime is a proprietary protocol meant for fast re-configuration of a ring topology (30ms re-configuration time). This protocol is also based on multicasting. FRNT is, however, not limited to the multicast addresses used by the FRNT control packets. Thus, standard IEEE1588 multicast packets can in principle be sent and received on a port in FRNT “BLOCKING” mode.

III. Layer 3 Network Redundancy Protocol

A layer 3 network redundancy protocol can also be used for the purpose of sending/receiving IEEE1588 packets on 100BASE trunk links only. Layer 3 is a better choice than layer 2 from a load balancing point of view since all links are used for sending/receiving packets (IEEE1588 and other packets can be sent on the same 100BASE link), but layer 3 means increased cost compared to layer 2. The router (layer 3) implementation must make sure that IEEE1588 packets are forwarded on 100BASE ports only if such ports are available. A layer 3 network topology can be made more complex than a corresponding layer 2 topology, but the user must still ensure that 100BASE paths exist between all IEEE1588 Slaves and the IEEE1588 Grand Master(s).

IV. VLAN and Multicast Filtering Techniques

An alternative approach is to use VLAN and special IEEE1588 multicast filtering techniques instead of using network redundancy protocols. The port definitions for an Ethernet switch with IEEE1588 Transparency support can be as follows

- *Timing Traffic Only (Blue Port):* Only incoming packets that are part of 1588 communications traffic will be accepted and forwarded to the switch CPU on a port configured as a "Timing Traffic Only (Blue Port)". The switch CPU can forward IEEE1588 packets on ports configured for "Timing Traffic Only (Blue Port)" or "All Traffic (Black Port)".
- *Non-Timing Traffic (Red Port):* All incoming packets except IEEE1588 packets will be accepted on a port configured as a "Non-Timing Traffic (Red Port)" and such packets can be forwarded to the switch CPU or ports configured for "Non-Timing Traffic (Red Port)" or "All Traffic (Black Port)".
- *All Traffic (Black Port):* IEEE1588 packets received on a port configured for "All Traffic (Black Port)" will be handled as if the IEEE1588 packets were received on a port configured for Timing Traffic Only (Blue Port). Non IEEE1588 packets will be handled as if the packets were received on a port configured for "Non-Timing Traffic (Red Port)"

The above port settings can be achieved by using port based VLAN techniques and special principles for the IEEE1588 multicast filters. I.e. port based VLAN’s can be used for port isolation and the IGMP snooping implementation will in case of the IEEE1588 multicast groups depend on the above time port settings.

A “Timing Traffic Only (Blue Port)” will be a 100BASE trunk port, the “Non-Timing Traffic (Red Port)” can be a 100BASE or 1000BASE trunk port, while an “All Traffic (Black Port)” can be a 100BASE or 1000BASE non-trunk or trunk port.

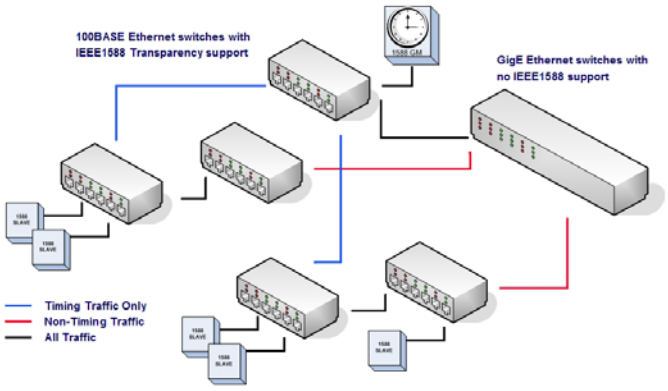


Figure 2, VLAN and multicast techniques

V. Summary

Accurate IEEE1588 time sync in a 1000BASE network can be done even though the 1000BASE switches have no support for IEEE1588. The network implementation can be based on the following:

- Utilize that 10/100BASE is used on non-trunk ports and 1000BASE on the trunk ports.
- Add extra 100BASE trunk ports used for time sync distribution, i.e.:

1

Migration towards IEEE1588 time synchronization over gigabit Ethernet

Sven Nylund
sven@ontimenet.com

Øyvind Holmeide
oeyvind@ontimenet.com

westermo OnTime

2

Migration towards IEEE1588 over GigE

Agenda:

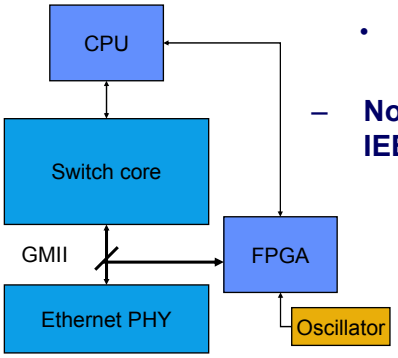
- **The GigE/time sync challenge**
- **Intermediate solution**
- **Alternative implementations:**
  - **Layer 2 network redundancy protocol**
  - **Layer 3 network redundancy protocol**
  - **VLAN and multicast filtering techniques**

westermo OnTime

3

Migration towards IEEE1588 over GigE

- **The GigE/time sync challenge:**
  - **The PHY/MAC GMII interface is a high speed interface that requires:**
    - **Expensive FPGA solution**
    - **Proper impedance matching**
  - **No commercial GigE chipset with IEEE1588 support available yet.**



westermo OnTime

4

Migration towards IEEE1588 over GigE

- **Intermediate solution:**
  - **Utilize that 10/100BASE is used on non-trunk ports and 1000BASE on the trunk ports**
  - **Add extra 100BASE trunk ports used for time sync distribution**
  - **I.e.: no IEEE1588 time sync packets are sent through the GigE switches that have no support for IEEE1588**

westermo OnTime

5

Migration towards IEEE1588 over GigE

- **Intermediate solution cont'd:**
  - **Extra 100BASE trunk links will create loops**
  - **Network loops must be handled in order to avoid broadcast storms**

westermo OnTime

6

Migration towards IEEE1588 over GigE

- **How to avoid network loops:**
  1. **Layer 2 network redundancy protocol**
  2. **Layer 3 network redundancy protocol**
  3. **VLAN and multicast filtering techniques**

westermo OnTime

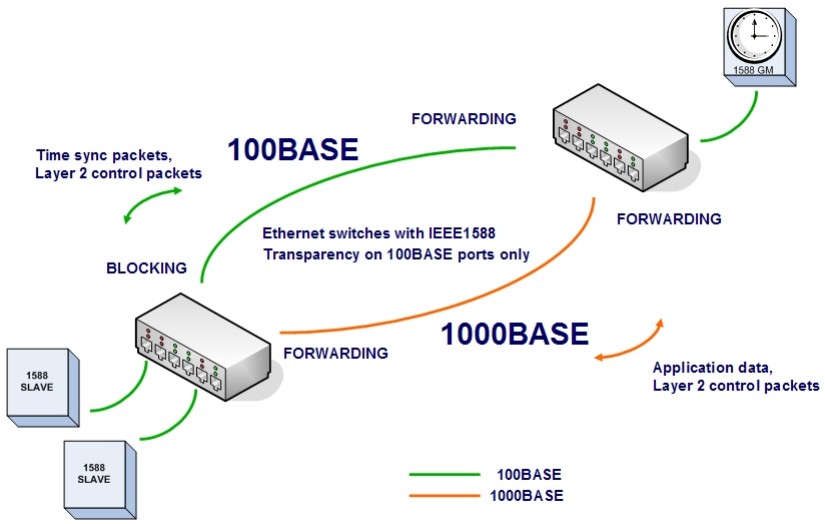
Migration towards IEEE1588 over GigE

1. **Layer 2 network redundancy protocol:**
   - **Spanning Tree (STP/RSTP - IEEE802.1D/ IEEE802.1w)**
   - **Fast Re-configuration of network Topology (FRNT)**

westermo OnTime

Migration towards IEEE1588 over GigE

- **1. Layer 2 network redundancy**



Migration towards IEEE1588 over GigE

1. **Layer 2 network redundancy protocol cont'd:**
   - **STP/RSTP**
     - **IEEE1588 packets cannot be sent on ports in BLOCKING mode**
     - **User configuration required**
     - **Hard to predict possible spanning trees for complex network topologies**
   - **FRNT:**
     - **IEEE1588 packets be sent on FRNT ports in BLOCKING mode,**
     - **Westermo OnTime propritary protocol**

Migration towards IEEE1588 over GigE

2. **Layer 3 network redundancy protocol:**
   - **OSPF**
   - **VRRP**
   - **L3 multicast filtering**
   - **Load balancing**

Migration towards IEEE1588 over GigE

2. **Layer 3 network redundancy protocol cont'd:**
   - **Can be used for complex topologies as long as 100BASE network paths exist**
   - **IEEE1588 and normal traffic can be combined**
   - **Expensive**

westermo OnTime

Migration towards IEEE1588 over GigE

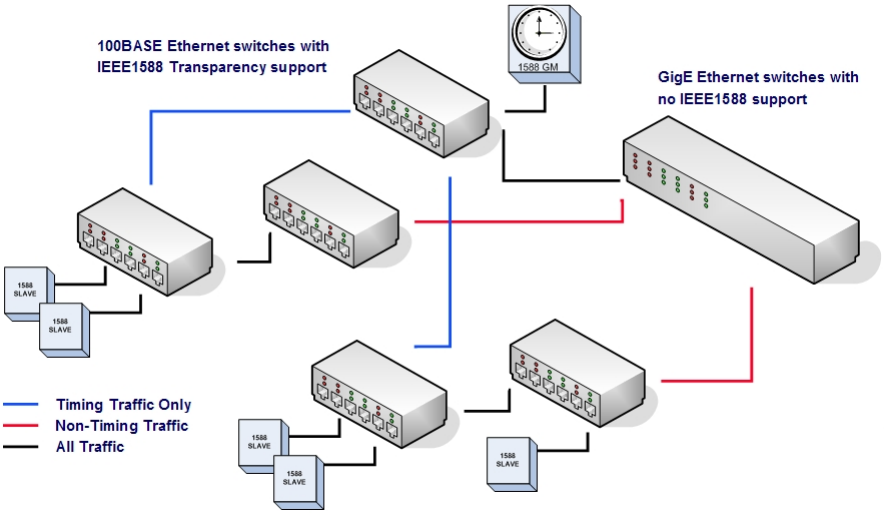
3. **VLAN and multicast filtering techniques, port definitions:**
   - **Timing Traffic Only (Blue Port)**
   - **Non-Timing Traffic (Red Port)**
   - **All Traffic (Black Port)**

westermo OnTime

13

Migration towards IEEE1588 over GigE

VLAN and multicast filtering techniques:



14

Migration towards IEEE1588 over GigE

3. VLAN and multicast filtering techniques, cont'd:

- User configuration required
- Management (SNMP, etc) must be possible via "Blue ports"
- Can be combined with network redundancy protocols

westermo OnTime

Migration towards IEEE1588 over GigE

Lynx1400 GigE switch
without IEEE1588 support

T200 with IEEE1588 Grand Master
or Transparency support

Summary:

- **Accurate IEEE1588 time sync in a GigE network can be done even though the GigE switches have no support for IEEE1588, i.e.:**
  - **Utilize that 10/100BASE is used on non-trunk ports and 1000BASE on the trunk ports**
  - **Add extra 100BASE trunk ports used for time sync distribution, i.e.:**
  - **No IEEE1588 time sync packets are sent through the GigE switches that have no support for IEEE1588**

westermo OnTime

Clock Synchronization based on Transparent Clock Approach

Author: Dr. Matthias Wenk
Address: Siemens AG, Division Automation and Drives
Frauenauracher Strasse 80
91056 Erlangen, Germany
E-Mail: matthias.wenk@siemens.com

Abstract:

This paper is discussing the industrial automation requirements on synchronization in switched Ethernet networks. A new clock type – Transparent Clock – is introduced to meet these requirements. Additionally a new delay measurement protocol is presented. This protocol is called ADelay protocol. It enables fast start-up and fast network reconfiguration phases.

1 Purposes of Synchronization for Industrial Automation

Industrial automation networks are distributed systems. For a lot of applications the synchronization of distributed nodes to a common time base is very important. A common time base is needed e.g. for time stamping diagnosis and process events. Based on these timestamps the relation of cause and effect is determinable. This helps the user to find out the reason for malfunction of the network or the process.

Also important is the ability of coordination of common activities. In motion control applications drives have to be synchronized to each other. This is important e.g. to guarantee the printing accuracy of printing machines.

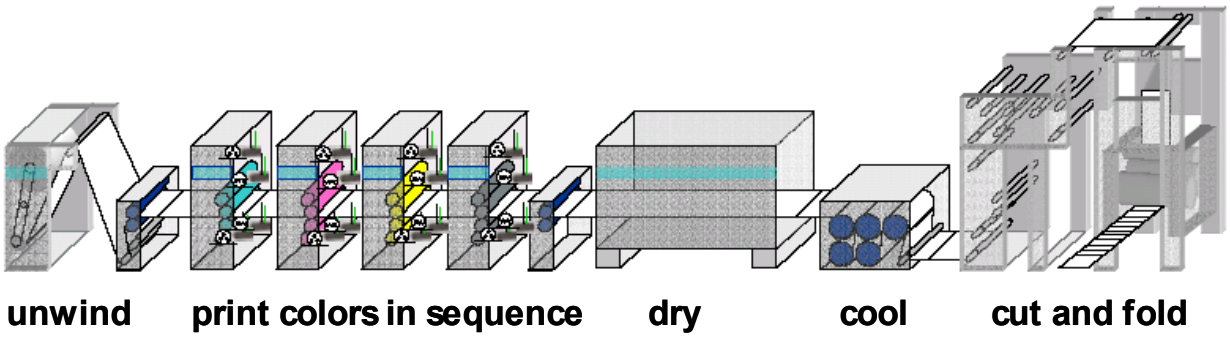


Figure 1: Synchronized drives within a four color offset printing machine

A four color printing machine is printing the colors in sequence (Figure 1). Thereby it is important that the deviation of the printing position is not to high because this results in a bad printing result. A short example shows the requirements on the synchronization accuracy. With a paper speed of about 20 meters per second and a required printing

accuracy of ±5 micrometers a synchronization accuracy of ±250 nanoseconds is needed.

2 Used Topologies

Modern industrial automation networks are based on 100 Mbit/s full duplex switched Ethernet according to IEEE 802.1D [1]. To ensure quality of service (QoS) packet prioritization [2] and time division multiplexing (TDM) at Layer-2 are additionally needed if spontaneous IT traffic is also transferred via the automation network. In general all topologies can be used by industrial automation networks. That can be a star topology, a tree topology, a loop topology or a daisy chain. But loop topologies and daisy chains are preferred. The loop topology is very important for the reliability of Ethernet based networks. If an Ethernet node in a daisy chain is breaking down then the nodes behind it are no more accessible. This could be avoided by using loop topologies. With a loop topology there is always a redundant path via messages could be transferred. This increases the reliability significant. Thereby it is called a media redundant topology.

3 Effects on using Switch Technology

Using switched Ethernet has a lot of advantages against shared media Ethernet. There are no collisions of packets on the wire. Collisions reduce the deterministic quality of a network. Determinism is one the most important requirements on industrial automation. Another advantage of switched Ethernet is the ability of full duplex transmission. This enables a simultaneous network traffic between different nodes which is not possible with shared Ethernet. And last but not least switched Ethernet is state of the art and will be enhanced furthermore.

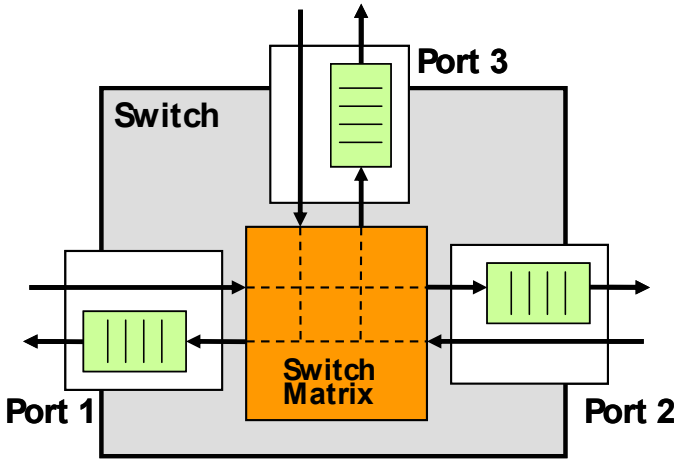


Figure 2: Model of an Ethernet bridge (switch) according to IEEE802.1d

But on the other hand the use of switches complicates the synchronization task. Collisions on the wire are avoided by queuing the messages inside the switches. If the

output queue of a switch is not empty while forwarding a message to this port, the message has to be queued (Figure 2). So the residential time of the message inside the switch depends on the network load. Also if the output queue is empty the residential time is in the range of some microseconds and has to be considered. For comparison: Using a shared media the residential time within a hub is about 500 nanoseconds.

Because of the variation of the residential time of messages within a switch the propagation delay of PTP messages is also varying along a daisy chain of switches. Synchronizing a switched Ethernet the residential times within switches between Master and Slave Clocks could not be ignored!

The existing IEEE1588 standard [3] includes a clock type called Boundary Clock. With Boundary Clocks the synchronization of distributed network nodes is reduced to a point-to-point synchronization (Figure 3).

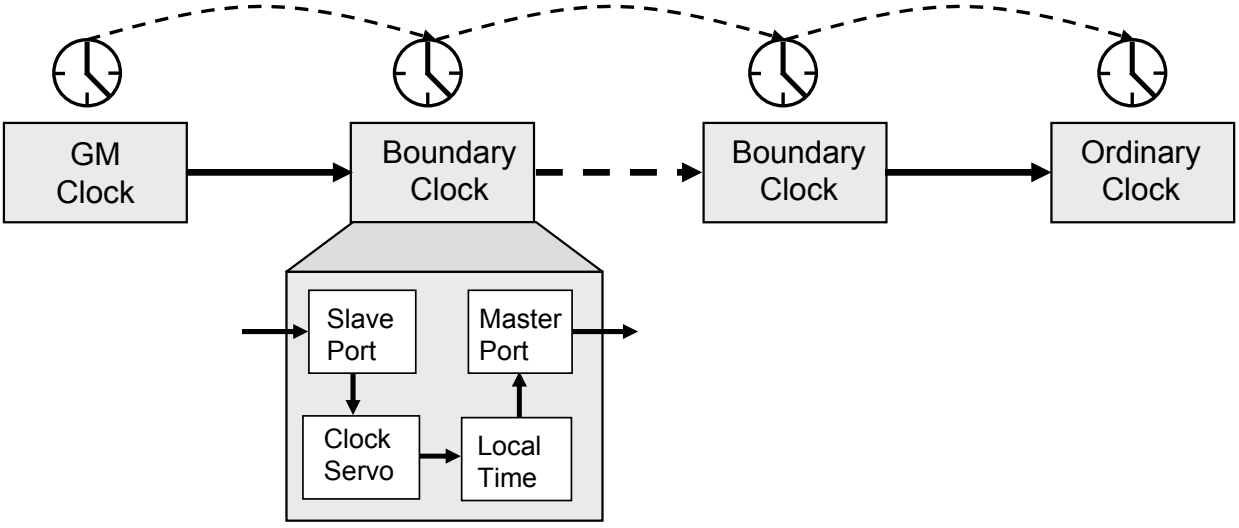


Figure 3: Boundary Clocks in a daisy chain topology

A Boundary Clock receives Sync/Follow_Up messages via one slave port. The slave port is determined by a Best Master Algorithm (BMC). The Sync/Follow_Up messages are used to synchronize the local time of the node. Every Boundary Clock itself acts as a Master Clock. It conveys Sync/Follow_Up messages, based on the local time, via its master ports. If Boundary Clocks are linked to a daisy chain this results in a cascading of servo control loops. Accuracy and stability issues are caused. As a result, Boundary Clocks are not suitable for daisy chain topologies.

4 New Clock Type: Transparent Clock

As discussed in section 2, daisy chain and loop topologies are preferred in industrial automation networks. For using these topologies in a switched network a new clock type is needed.

4.1 Introduction

The new clock type is called Transparent Clock. The main difference to Boundary Clocks is the forwarding of Sync and Follow_Up messages. As a result every Ordinary Clock is synchronized to the Grandmaster. A cascading of control loops could be avoided. But Transparent Clocks are not only forwarding nodes between Grandmaster Clock and Ordinary Clocks. A Transparent Clock could also act as an Ordinary Clock (Figure 4).

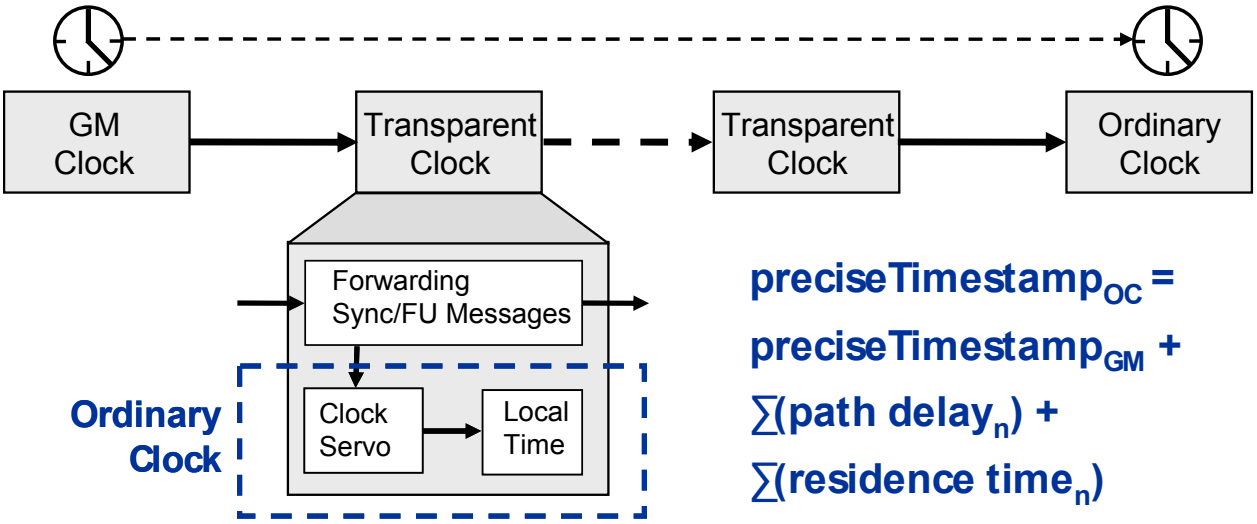


Figure 4: Transparent Clocks in a daisy chain topology

The Sync and Follow_Up messages are conveyed directly from Grandmaster Clock to Ordinary Clocks. To determine the precise timestamp within an Ordinary Clock the propagation delay of the messages on the transmission path is needed. The propagation delay is the sum of all path delays and residence times along the transmission path between Grandmaster Clock to Ordinary Clock. Because of alternating residence times in the switches the propagation delay could not be measured with the existing delay protocol.

4.2 New Delay Measurement Protocol - ADelay

The Transparent Clock approach uses a new kind of delay measurement protocol called ADelay protocol. The difference to the existing delay measurement protocol is that the transmission delay is not measured “end-to-end” between a Grandmaster Clock and an Ordinary Clock. It is measured between two neighbored nodes only (Figure 5). The

delay measurement is not triggered by the reception of a Sync message. It can be initiated by the Delay Requestor at any time. A Delay Requestor is initiating it at every of its ports. The Delay Requestor is time stamping the transmitting point of time (T_1) of the ADelay_Req message. The Delay Responder is time stamping the receiving point of time (T_2). T_2 is conveyed to the Delay Requestor via the ADelay_Resp message. The Delay Requestor is time stamping the receiving point of time (T_4) of the ADelay_Resp message. The transmitting point of time (T_3) of the ADelay_Resp message is conveyed to the Delay Requestor via the ADelay_RespFU message.

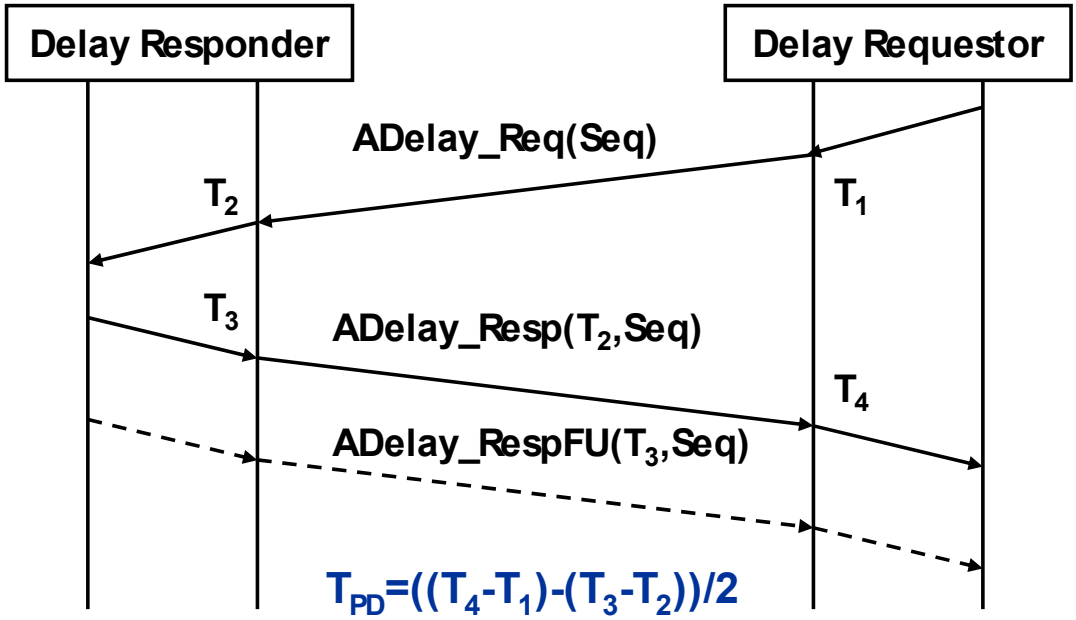


Figure 5: Timing diagramm of ADelay protocol

The path delay T_{PD} between the Delay Requestor and the Delay Responder can be calculated with equation (1):

$$T_{PD}=((T_4-T_1)-(T_3-T_2))/2 \quad (1)$$

A significant advantage of the ADelay protocol is that Delay Requestor and Delay Responder do not have to be synchronized, because only local time differences are used to calculate the path delay.

4.3 Functionality of Transparent Clocks

As discussed in section 4.1, the propagation delay is the sum of all path delays and residence times along the transmission path between a Grandmaster Clock and an Ordinary Clock. Because of alternating residence times in the switches the residence time of a Sync message has to be measured with every message. Hence the path delay and the variable residence time are added by every switch to a specific field within the Sync or Follow_Up message (Figure 6).

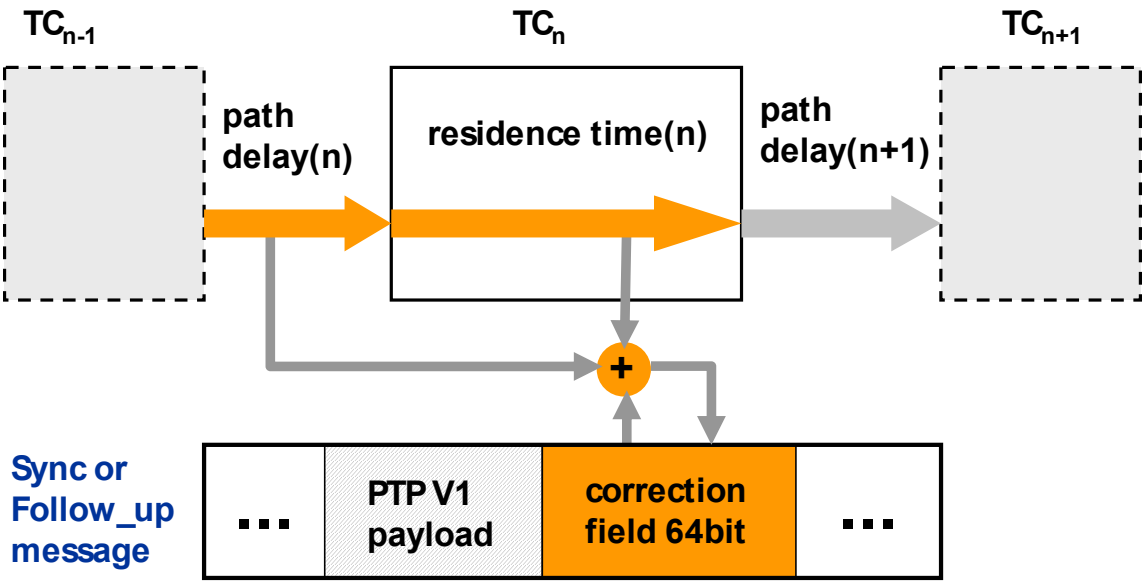


Figure 6: Adding path delay and residence time to correction field.

The correction field is an extension field to the existing PTP payload. Only if the path delay to the neighbored node is known the Sync message can be forwarded. So the path delay is measured by ADelay protocol after the change of a port state to link up state and afterwards periodically. This shortens the start-up phase significant, because the delay measurement has not to be delayed till the first reception of a Sync message.

There are two kinds of Transparent Clocks defined:

- Change-on-the-fly Transparent Clock
- Follow_Up Transparent Clock

A change-on-the-fly Transparent Clock is able to manipulate the correction field while sending a Sync Message. This feature requires hardware support. An additional Follow_Up message is not needed. This enables decreasing the network load.

A follow_up Transparent Clock is time stamping the receiving and the transmitting point of time of the Sync message but keeps the Sync message unchanged. The calculated residence time is added to the correction field of the Follow_Up message.

5 Transparent Clocks and Media Redundancy

Transparent Clocks are suitable to use in media redundant topologies. The substantial requirement on media redundancy is to minimize the duration of the reconfiguration phase. To meet this requirement the use of ADelay measurement protocol and the forwarding of Sync/Follow_Up messages is needed.

Within a loop topology the circulation of messages has to be avoided. This is done by disabling (or blocking) one port of one node within the loop (Figure 7). Spanning Tree

Algorithm [4] could be used to disable a port within a loop topology. PTP messages will not be forwarded via a disabled port. So, also the ADelay measurement could not be executed.

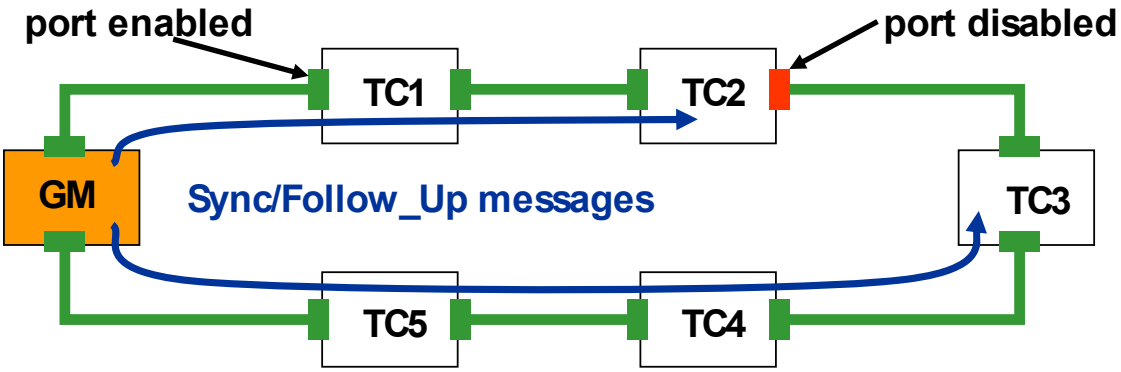


Figure 7: Avoiding message circulation by disabling one port in the loop

In the case of breaking the loop the disabled port in TC2 will be enabled. This could be done by a Spanning Tree Algorithm [4]. As a result the Transparent Clocks TC3, TC4 and TC5 receive the Sync and Follow_Up messages via the opposite direction (Figure 8). After the ADelay measurement between the Transparent Clocks TC2 and TC3, the Sync and Follow_Up messages could be forwarded immediately by every Transparent Clock because the path delay is available in every node.

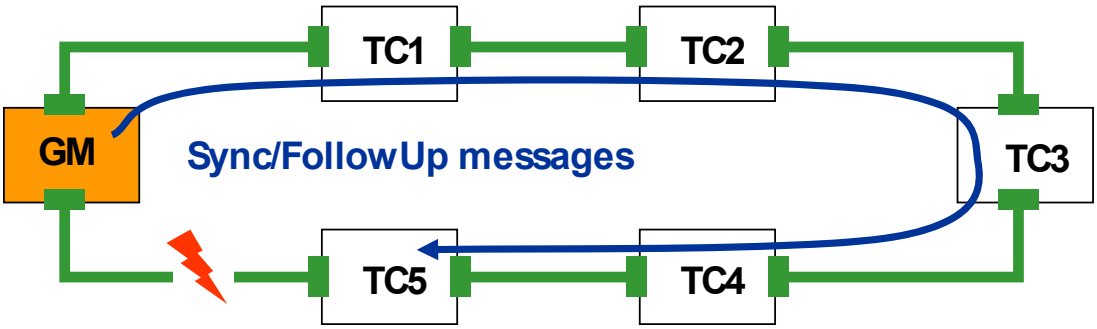


Figure 8: Enabling the disabled port if the loop is broken

If the loop topology consists of Boundary Clocks every Boundary Clock has to do at first a new delay measurement after receiving the first Sync message via the redundant path. This is done sequentially with every Boundary Clock. This behavior extents the reconfiguration phase after breaking a loop.

As a result the Transparent Clock approach and the ADelay protocol are needed to minimize the reconfiguration phase and also the start-up phase.

6 Summary

Industrial Automation requires non-cascading of clock servos with daisy chain topology as well as fast reconfiguration phases with loop topologies.

Transparent Clocks meet these requirements by forwarding Sync and Follow_Up messages and by using a new delay measurement protocol (ADelay protocol). The path delay to the neighbor node and varying residence time are added to an additional correction field within the Sync or Follow_up message.

7 References

[1] ANSI/IEEE Std. 802.1D-2004: IEEE Standard for Local and Metropolitan Area Networks - Media Access Control (MAC) Bridges, IEEE, New York, 2004

[2] ANSI/IEEE Std. 802.1Q-2003, IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks, IEEE, New York, 2003

[3] ANSI/IEEE Std. 1588-2002, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE, New York, 2002

[4] RFC 1157 - Simple Network Management Protocol (SNMP), RFC Editor, 1989, http://www.rfc-editor.org/

Clock Synchronization based on Transparent Clock Approach

Dr. Matthias Wenk
Siemens AG, Automation and Drives
Erlangen, Germany

SIEMENS

Outline

- Synchronization purposes of industrial automation
- Used topologies in switched automation networks
- Introduction of new Clock Type: Transparent Clock
- New delay measurement protocol - ADelay
- Transparent Clocks and media redundancy

SIEMENS

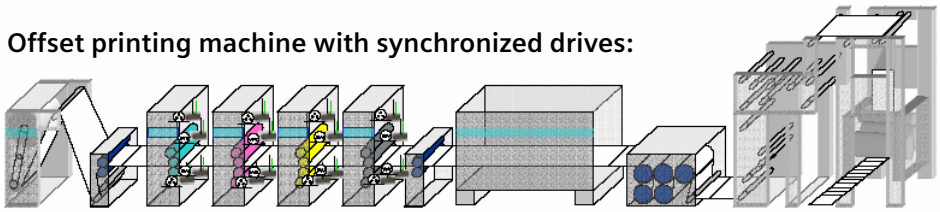
Synchronization of Industrial Automation Networks

Transparent Clock Approach
IA requirements
Boundary Clock
Transparent Clock
Delay Measurement
Media Redundancy
Summary

A common time base is needed in industrial automation for:

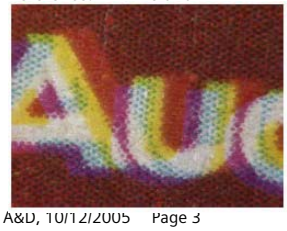
- Time stamping of diagnosis and process events
- Determine relation of cause and effect (causation)
- Coordinating common activities (synchronization)

Offset printing machine with synchronized drives:



unwind print colors in sequence dry cool cut and fold

Reference: MAN Roland



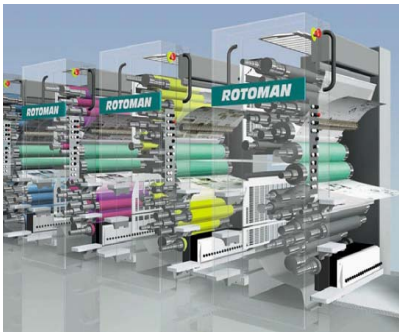
- speed up to v=20m/sec
- printing accuracy Δs=±5µm

→ synchronization accuracy: Δs/v = ±250ns

Application Example: Offset Printing Machine

Transparent Clock Approach
IA requirements
Boundary Clock
Transparent Clock
Delay Measurement
Media Redundancy
Summary

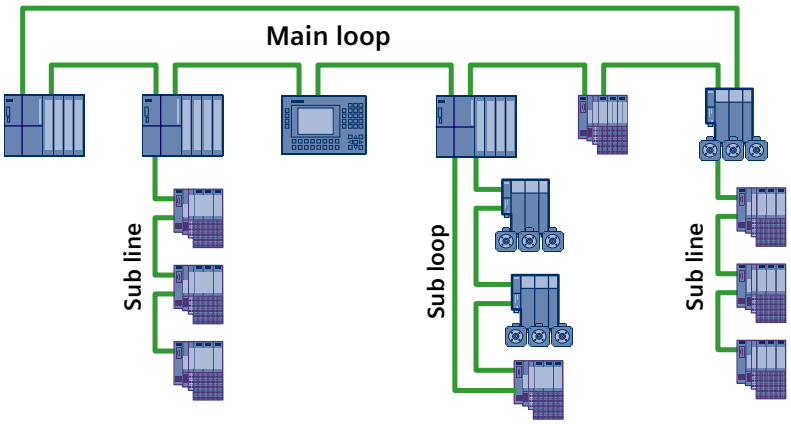
Reference: MAN Roland



Industrial Ethernet – Used Topologies

Transparent Clock Approach
IA requirements
Boundary Clock
Transparent Clock
Delay Measurement
Media Redundancy
Summary

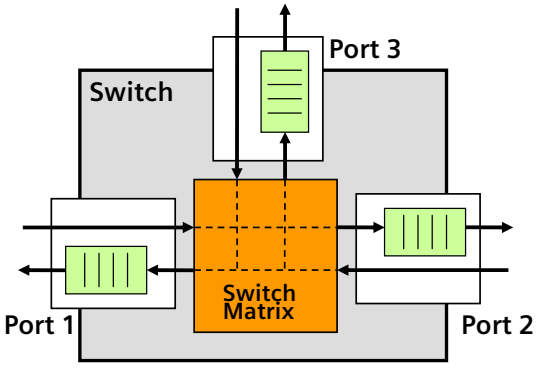
- In general all topologies can be used (star, tree, loop, daisy chain).
- Daisy chain topology is preferred. Main line is expanded by sublines.
- Usage of daisy chains is critical with switched ethernet networks.
- Daisy chains are often extended to a loop topology.
  Media redundancy increases reliability.



SIEMENS

Switched Ethernet: Effects on Synchronization

Transparent Clock Approach
IA requirements
Boundary Clock
Transparent Clock
Delay Measurement
Media Redundancy
Summary



Pros:
- No collisions on the wire
- Higher bitrates useable
- Full-duplex transmission
- Simultaneous network traffic

Drawback:
- Switch is a queuing system

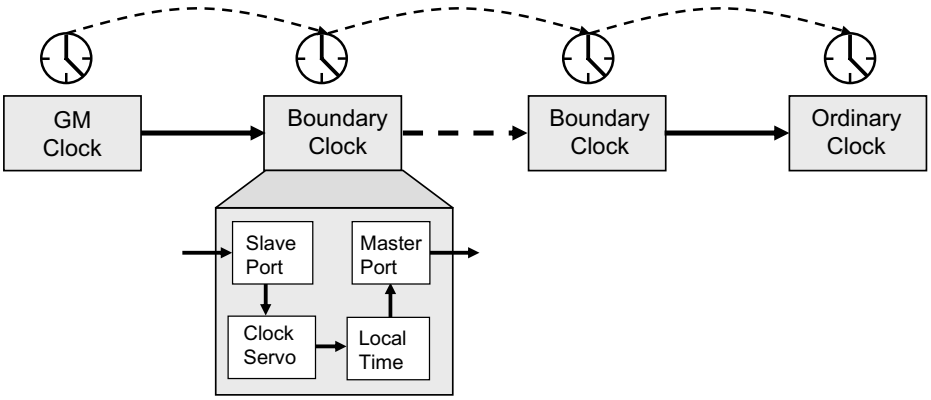
- Messages will be queued if the output port is busy.
- Residence time of messages depends on the load of the output queue. Thus it depends on the network load -> no constant delay!
- TDM and priority tagging are helpful but not a sufficient solution.
- Also if the output queue is empty, the residence time is higher than it is with a hub (500ns -> 3..125 µs)

→ For synchronizing a switched ethernet the alternating residence times of the switches could not be ignored!

SIEMENS

Boundary Clock

Transparent Clock Approach
IA requirements
Boundary Clock
Transparent Clock
Delay Measurement
Media Redundancy
Summary



Boundary Clocks solve the problem of alternating residence times by point-to-point synchronization.

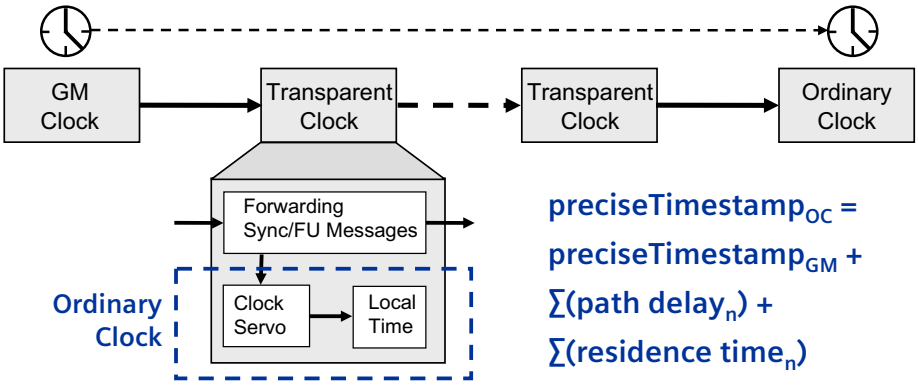
Issues:

- **Cascading of control loops → accuracy and stability problems**
- **Prolong start-up and reconfiguration phase.**

→ Boundary Clocks are not suitable for long daisy chains and media redundant topologies

New Clock Type - Transparent Clock

Transparent Clock Approach
IA requirements
Boundary Clock
Transparent Clock
Delay Measurement
Media Redundancy
Summary

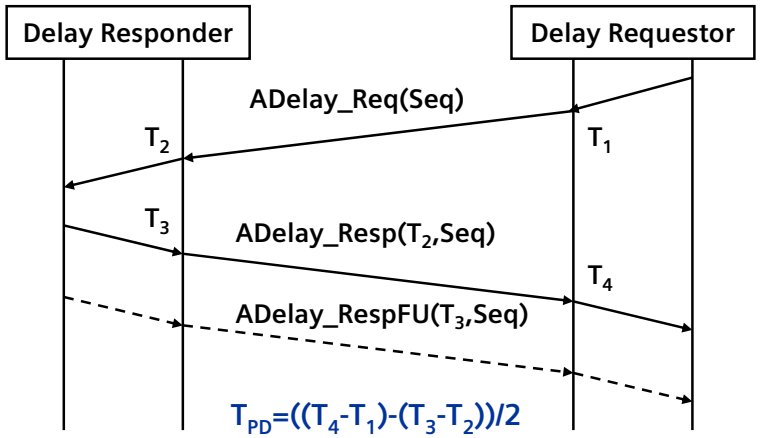


$preciseTimestamp_{OC} = preciseTimestamp_{GM} + \sum(path\ delay_n) + \sum(residence\ time_n)$

- **Sync/Follow_Up messages are forwarded by Transparent Clocks → Only one control loop between Grandmaster Clock and Ordinary Clock.**
- **Because of alternating residence times in the switches the propagation delay of Sync messages has to be measured.**
- **Propagation delay is the sum of all path delays and residence times along the transmission path between GM and OC.**
- **Transparent Clocks can also be synchronized. In this case they act as Transparent Clock switches with additional Ordinary Clock function.**

Delay Measurement with Transparent Clocks

Transparent Clock Approach
IA requirements
Boundary Clock
Transparent Clock
Delay Measurement
Media Redundancy
Summary

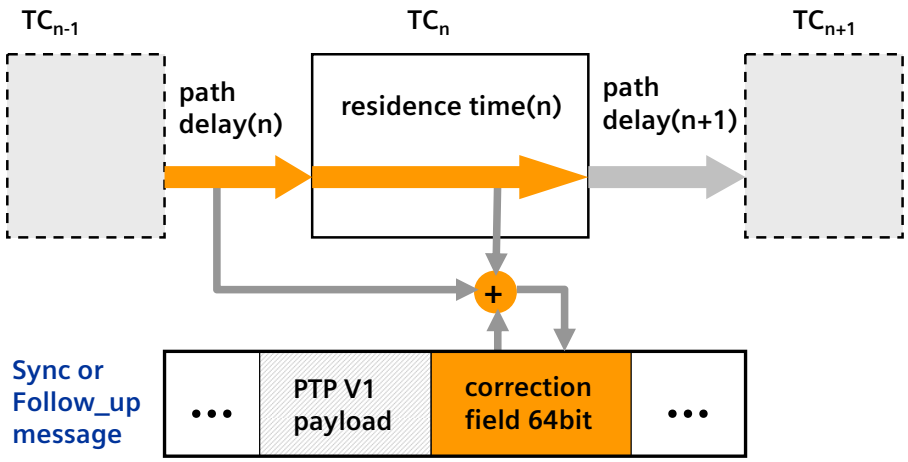


- ADelay measurement is done between neighbored TCs (P2P).
- A Transparent Clock does ADelay measurement at every port.
- ADelay measurement is not triggered by Sync message.
- ADelay measurement is a precondition for forwarding Sync messages.

→ ADelay measurement shorten start-up and reconfiguration phase.

Functionality of Transparent Clocks

Transparent Clock Approach
IA requirements
Boundary Clock
Transparent Clock
Delay Measurement
Media Redundancy
Summary

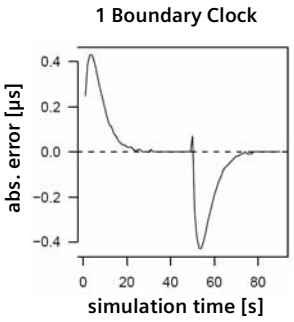


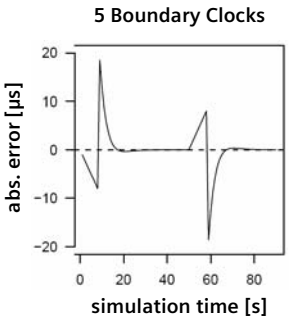
There are two kinds of Transparent Clocks:

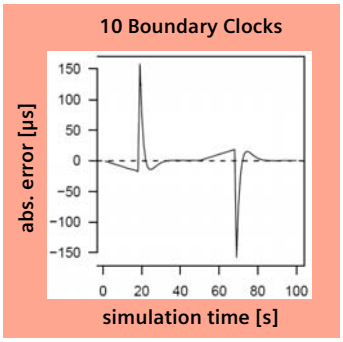
- „Follow-up Transparent Clock“: change correction field in Follow-up message
- „Change-on-the-fly Transparent Clock“: change correction field in Sync message while sending (Follow_Up message not needed but hardware support)

Simulation Studies

Transparent Clock Approach
IA requirements
Boundary Clock
Transparent Clock
Delay Measurement
Media Redundancy
Summary







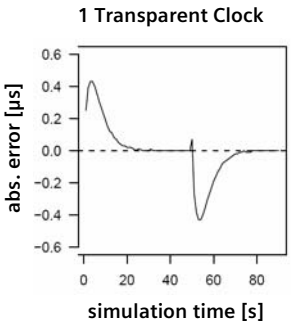
Zürcher Hochschule Winterthur

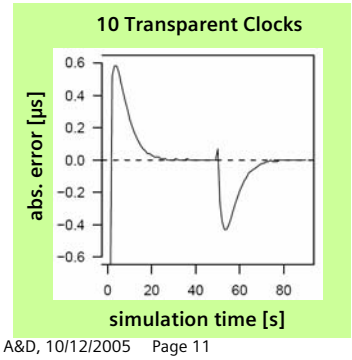
Clocks in daisy chain

Oscillator drift: 1ppm

PI drift controller

Sync interval: 1 second

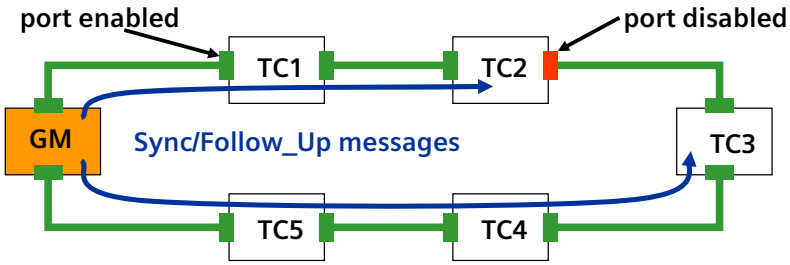




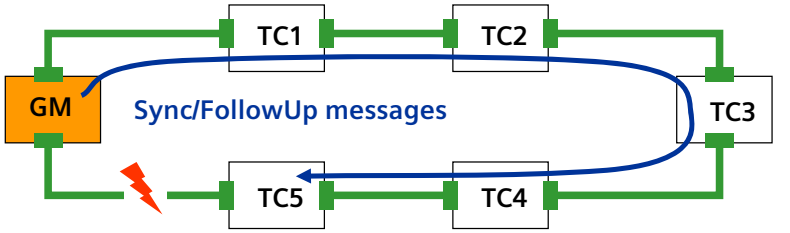
SIEMENS

Transparent Clocks and Media Redundancy

Transparent Clock Approach
IA requirements
Boundary Clock
Transparent Clock
Delay Measurement
Media Redundancy
Summary



Disabling of one port within a loop (STP, RSTP, others)



Change of transmission direction for TCs 3,4,5.

→ Fast reconfiguration and fast start-up requires ADelay protocol

SIEMENS

Summary

Transparent Clock Approach
IA requirements
Boundary Clock
Transparent Clock
Delay Measurement
Media Redundancy
Summary

- Industrial automation requires a synchronization mechanism to rule long daisy chains and loop topologies in switched ethernet networks.
- The new transparent clock approach was designed to meet exactly these requirements.
- The basic features of transparent clocks are
  - forwarding of Sync and Follow_Up messages and
  - consideration of path delays and residence times along the transmission path
- A new delay measurement protocol supports fast start-up and fast reconfiguration in media redundant topologies.
- Next step: Layer-2 transport

SIEMENS

Clock Phase Change compensation using Graham Scan

Augusto Ciuffoletti
Università di Pisa

Galina Antonova
GE Consumer & Industrial, Multilin

Problem statement

- Ensuring high timing accuracy for Slave clocks built with cheap oscillators requires frequent updates from the Master clock.
- Compensation of linear component of Clock Phase Change may significantly reduce updates' frequency, while maintaining required timing accuracy.
- Temperature and ageing (non-linear) components of the Clock Phase Change remain to be compensated.

Graham Scan requirements

- The algorithm requires that a Slave clock receives a sequence of timestamped messages from a Master.
- Although the message flow should be regular, no strict timeliness is required.
- Such messages do not need special privileges, but regularity in the delivery helps.

Probabilistic issues

- The algorithm makes (weak) hypotheses on the distribution of message latency.
- The algorithm returns an estimate of the Clock Phase Change, not the “real” value.
- Accuracy of such estimate improves for longer message sequences.
- The algorithm improves performance of a clock synchronization algorithm, but does not replace it.

Advantages

- Low cost/impact algorithm.
- Adequate to a wireless environment: Slave does not need to transmit, thus can save power.
- Adequate to a broadcast/multicast environment: one message serves multiple Slaves.
- May be sufficient (no additional clock synchronization needed) for applications that only measure time intervals (e.g. monitoring, accounting, debugging).

Basic notation

$$ts(rcv_i) - ts(snd_i) = \Delta_i + a * ts(snd_i) + b$$

where

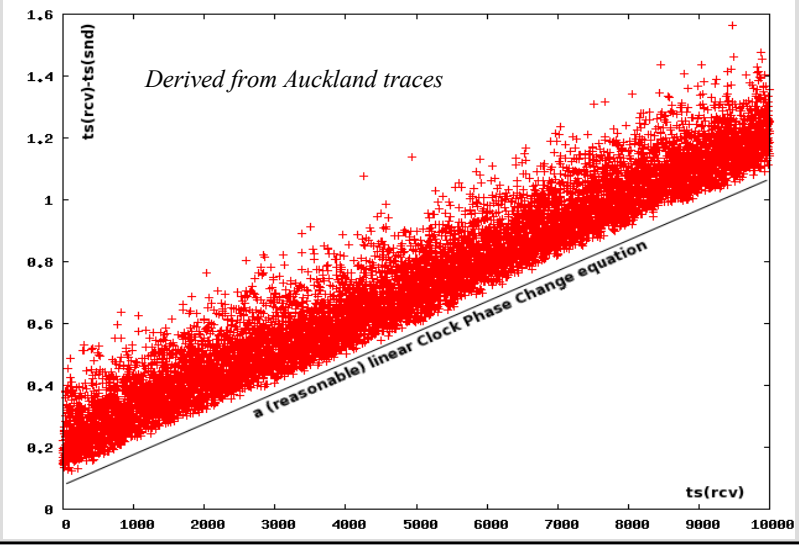
$ts(rcv_i), ts(snd_i)$ - Slave's receipt and Master's sending timestamps, based on their local clocks;

Δ_i - real message latency;

b - a linear component of the Clock Phase Change;

a - a constant component of the Clock Phase Change.

Timestamp difference plot

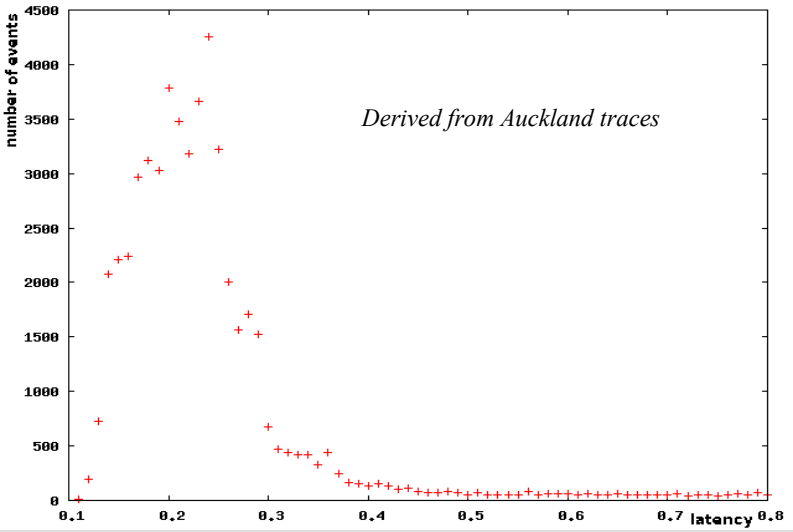


Clock Phase Change Interpolation

$$ts(rcv_i) - ts(snd_i) = \Delta_i + a * ts(snd_i) + b$$

- To compute a constant term of Clock Phase Change, two values for index *i* are required, such that two real message latencies are identical.
- Based on experience two minimal values of message latency in a sample are likely close.

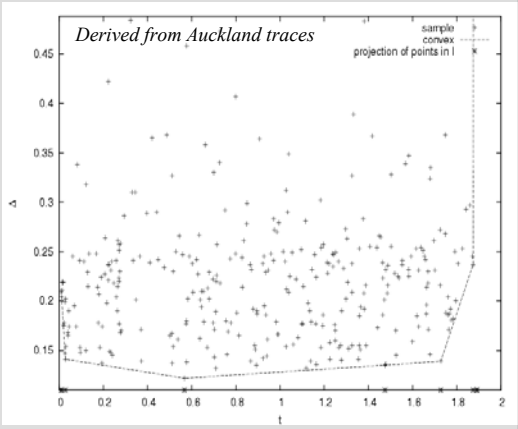
Message latency distribution



Finding minimal values

Points with minimal latency necessarily correspond to adjacent vertices of the (lower) polygon containing all points of latency graph.

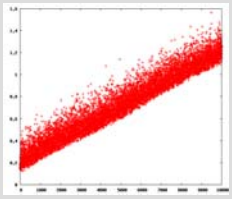
Proof: classical, by absurd.



Message Latency Graph

The Graham Scan

```
@hull->empty;
while <($snd,$rcv)> {
    $new=($snd,($rcv-$snd));
    while (test($hull(N),$hull(N-1),$new)){ pop @hull }
    push $new,@hull;
}
```



- the test function `test` computes and compares the slopes of the segments

(`$hull(N-1),$hull(N)`) and (`$hull(N-1),($snd,$rcv)`);

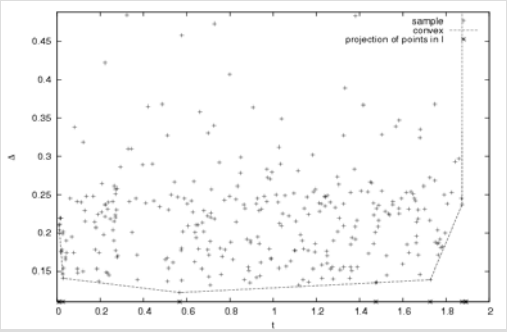
- the number of elements in `$hull` grows logarithmically with time.

Summarizing cost per time unit grows logarithmically with time.

Selecting minimal values

Selection rule:

select two successive edges in `$hull` , separated by the largest time gap.



- rationale:
  - minimizes worst case error of the estimate
  - easy to compute
- more investigation needed

Evaluating estimate accuracy

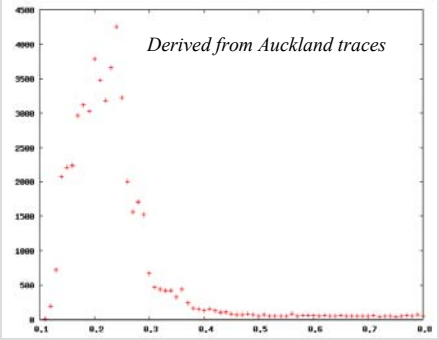
- Depends on communication delay distribution
- Increases with sample size
- May be affected by relevant non-linear (temperature driven) Clock Phase Changes
- May be affected by interfering clock adjustments

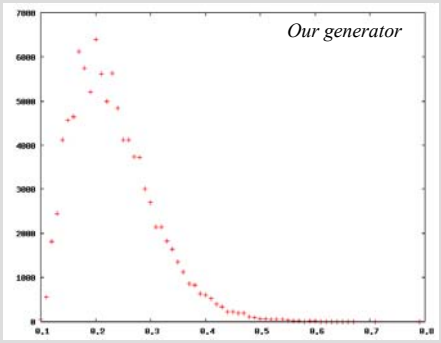
A simulation highlights long term aspects of the Clock Phase Change estimation algorithm.

Simulation basics

A key is our generator of one-way message latencies.

Our generator is fast and simulates long range dependence.



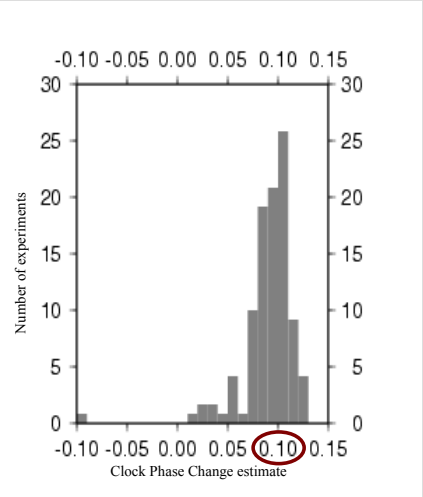


It is tuned on Auckland samples and introduces thermal variations.

Results: accuracy after stabilization

120 samples are generated with a Clock Phase Change of 0.1 parts per thousand (100 times better than a quartz clock) with periodic thermal shift.

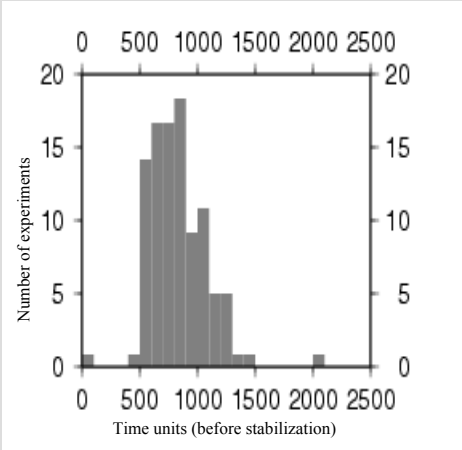
Estimate is read the first time the algorithm stabilizes (small variation of successive estimates).



Results: time to converge

Stabilization occurs when variation of successive estimates is small.

At 1 ping per second stabilization is reached after 20 minutes.



Conclusions

- Clock Phase Change compensation improves performance of clock synchronization.
- An efficient algorithm may significantly reduce Master to Slave updates' frequency, while maintaining required timing accuracy.
- Graham Scan algorithm offers an efficient low cost/impact solution.

Conclusions (continued)

- Clock Phase Change compensation using Graham Scan may bring savings in
  - Cost (by using cheaper oscillators),
  - Utilized bandwidth (by less frequent messages) and
  - Power consumption (by using one-way messages).
- Temperature/ageing component remains to be compensated.

References

- Moon, Skelley, Towsley “Estimation and Removal of Clock Skew from Network Delay Measurements”, TR98-43, Univ. of Massachusetts at Amherst.
- Cristian “Probabilistic Clock Synchronization”, Distributed Computing, 1989
- de Berg, van Kreveld, Overmars, Schwarzkopf *Computational Geometry*, pag 1-8, Springer 98.
- http://search.cpan.org/~augusto/Time-Skew-0.1/Skew.pm

A SIMULATION STUDY - PERFORMANCE OF IEEE 1588 OVER ETHERNET AND IEEE 802.11B WLANs

Dr. Nirmala Shenoy*, John Fischer**, Paul Myers** and Zeping Qui**
* Information Technology Department, Rochester Institute of Technology, **Spectracom Corp.
Rochester 14623, New York, USA

Abstract: IEEE 1588 proposes The Precision Time Protocol (PTP) to synchronize system clocks to sub microseconds accuracy; to be used in 802.3 based Ethernets. In this project, we conducted detailed simulation studies using the Opnet simulation tool on the PTP clock accuracy achieved in 802.3 Ethernet, with different types of LAN devices i.e. hubs and switches as part of the network configuration. We extended the studies to 802.11b WLANs, operating in the infrastructure and ad hoc modes. The tests were conducted under varying load conditions. In each test case, the offset and variance in the client clock due to traffic load, types of devices and network configuration were determined. The experiments were conducted by time-stamping the PTP frames both at PTP layer and at physical layer, just before the frames leave the PTP device, which helped study effects due to protocol and layer processing delays at PTP devices. Experiments to better understand effects of contention based medium access and back-off in the two LAN configurations were conducted.

INTRODUCTION

IEEE 1588 PTP standard was developed to provide sub microsecond accuracy in system clock synchronization in 802.3 Ethernet. However the synchronization accuracy achieved is dependent on the configuration of 802.3 Ethernet, i.e. the type of network devices, the number of devices in the network that contribute to the load, the processing delays and the point of time-stamping of the frames at the server and client. This joint project between RIT and Spectracom, Rochester, aims at investigating in detail the effects of the above mentioned factors in both 802.3 Ethernet and 802.11b WLANs. The study was extended to 802.11b WLANs due to recent increased interest in *Wireless Sensor Actuator Networks* for industry control, where high accuracy in time synchronization is essential. The studies were conducted using Opnet simulation tool and test beds.

For studies in both 802.3 Ethernet and 802.11b WLAN, extensive test cases were covered including different types of network devices, varying traffic loads and network configurations. Test cases to study effects of contention based medium access, collisions and back-off, and where applicable asymmetric traffic were included. Protocol and layer processing delays at PTP server and client were incorporated. Tests were conducted to study effects with and without processing delays i.e. by time-stamping the packets at PTP layer, where packets incurred delays due to PTP, MAC and other intermediate layer processing and just before they left the physical layer, where there are no delays. The investigative studies gave a clear insight into factors affecting synchronization accuracy achieved between server and client in such networks, which helped define application limitations of PTP clock and identify possible solutions to improve accuracy.

TERMINOLOGY

In our simulation models we time-stamped PTP packets at various stages of processing to study the system processing delay effects. We use figure 1 to explain certain terminology used in this context.

Time stamping outside MAC layer: In this case PTP packets are time stamped at the PTP layer and hence include protocol and processing delays at the PTP and other intermediate layer. All protocol processing delays were lumped together and modeled using a single M/G/1 queue process model.

Time stamping inside MAC layer: In this case PTP packets are time-stamped just before they leave the physical layer; hence protocol and layer processing delays are not incurred. If there is a retransmission, then the time stamp, when the packet was last sent is recorded. At the receiving end we assume that the PTP packets are time-stamped as soon as they are picked up at the physical layer.

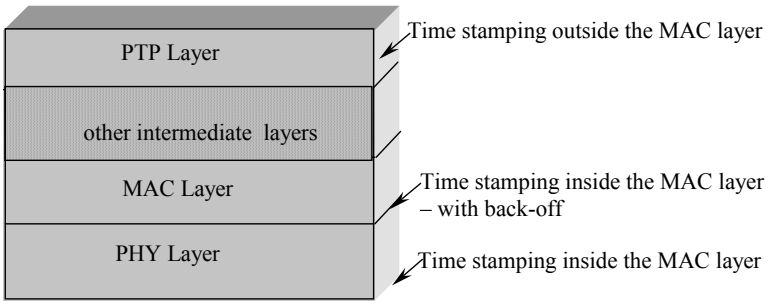


Figure 1 Time stamping points

Time stamping inside MAC layer- with back-off: In wireless network scenarios, we included one more time-stamping option to better study and understand the back-off effects. We did not include the protocol processing delays but if there is a retransmission we recorded the time when the first transmission was sent.

SIMULATION SETUP

In figure 2, we show the Opnet model used for the loaded hub network, where in 802.3 Ethernet LAN, 8 workstations are connected to a hub. Out of the 8 workstations, one of them is a PTP server and another is a PTP client. At the PTP devices, we modeled a PTP protocol, which runs directly on 802.3 MAC layer. The PTP server generates 'sync' packets every 1 second. When it receives a 'delay_request' packet from a client, it generates a 'delay_respone' packet. The PTP client generates a 'delay_request', when it receives the 'sync' packet from the server. Due to the simulation approach we were able to set the client clock to be exactly synchronized to the server clock.

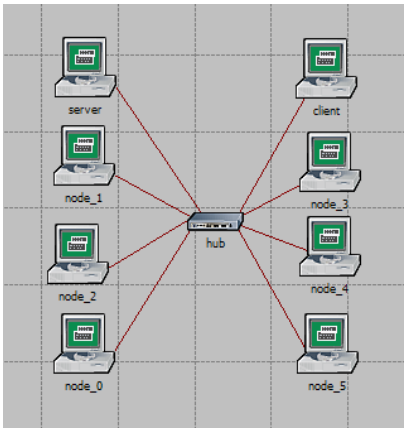


Figure 2 : Opnet model

The offset graphs provided below show instantaneous offset calculated at the client and the averaged offset over data collected. The simulation runs were repeated for various seeds. To simulate the load, a pseudo random number generator randomly generates traffic. When this generator is seeded with different starting values, the offset may vary considerably due to the back off effects and the processing delays.

PROTOCOL PROCESSING DELAY

The processing delays due to various protocol layers (from PTP to MAC layers as shown in figure 1) were lumped and modeled into one M/G/1 queue process. The adjacent text box gives the details of the queue model.

In the following sections, we first handle the performance achieved in hub based 802.3 Ethernet, followed by 802.3 switched Ethernet. We then handle various ad hoc and infrastructure cases under 802.11b WLAN. Our studies focus on estimating offset error at the client due to network effects. In the different scenarios, we always evaluated a test case with 'no load' condition to study the minimum delays in the network and its components.

The M/G/1 queue model is defined by the equation
$\Gamma = 1/\mu + \omega$, Γ is the total system waiting time,
μ is service rate, ω is queue waiting time.

$\omega = \eta(1/\mu^2 + \sigma^2)/2(1 - \eta/\mu)$, η is arrival rate for jobs at the queue, σ^2 is the variance.

For
$\sigma^2 = .0001$ ms^2, $\eta = .6$ to.95 job/ms, $\mu = 1$ job/ms
Γ(maximum) = 10.5 ms and Γ(minimum) = 1.75 ms

HUB BASED 802.3 ETHERNET

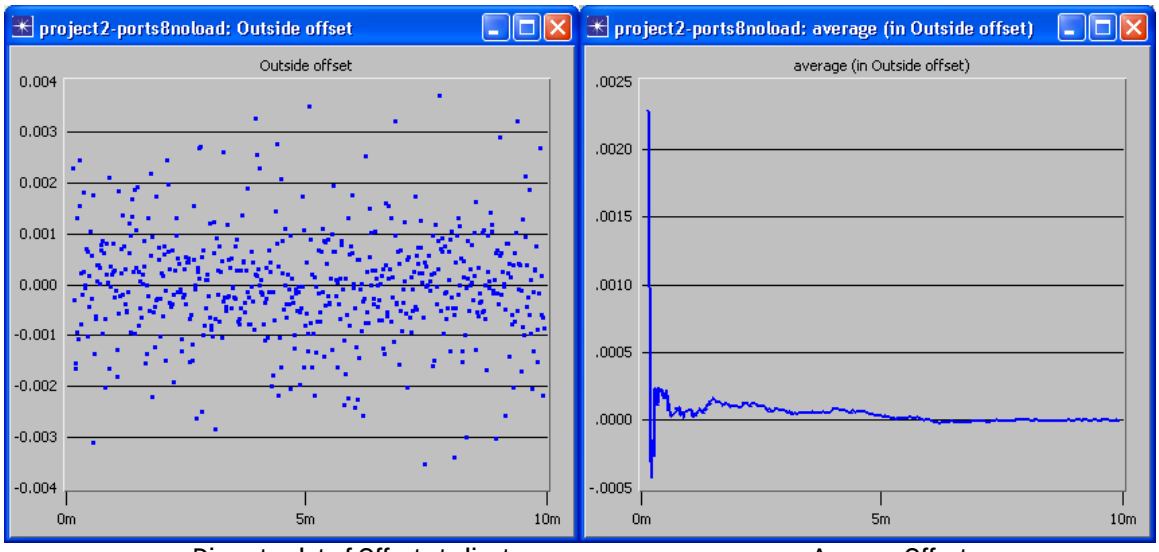
NO-LOAD HUB NETWORK

Time stamping inside the MAC layer: The delay calculated at the client was 5.9366 µs and the offset was 0. The delay calculated includes transmission delay, propagation delay, and hub delay. The adjacent text box provides the calculations used to determine the hub delay.

HUB DELAYS: 100BaseT Ethernet	
Packet size	= 576 bits
Transmission delay	= 5.76 µs
Propagation delay for link length of 37.5 meters	= 0.125 µs
Hence hub delay	= 51.6 ns[1].

Time stamping outside the MAC layer: The network delays in this case include the delays due to protocol and layer processing, and are highly variable. However, we focused primarily on the offset calculated at the client, as this factor gives the error in client clock synchronization due to network factors.

The averaged offset was 79.1258 µs and the variance was 1.272 ms[2]. The graphs below show the offset between server and client when simulated for 10 minutes. As the graph illustrates, the averaged offset approaches 0 with time.



Discrete plot of Offset at client

Average Offset

Inference: Even if there is no load in the network, the protocol processing delays can significantly affect synchronization achieved with PTP, in a hub-based network. However this effect depends on latency due to processing at the devices, which is variable. Averaging the offset at client reduces synchronization errors considerably. However 79.1258 µs is not in the sub-microseconds range.

LOADED HUB NETWORK

Time stamping inside the MAC layer: The data obtained under this case was the same as before, because contention and back-off will not affect the time-stamping since we assume the PTP packets are time-stamped just before they leave the physical layer and as soon as they are received.

Time stamping outside the MAC layer: The readings recorded for various loads are given in the table below.

Load	4.5%	9%	20%	45%	70% (seed 99)	70% (seed 1177)
Offset in µs	41.208	17.95	364	397	-100	-650

Inference: Whether loaded or unloaded, the synchronization obtained with time-stamping inside the MAC layer is highly accurate. With time stamping outside the MAC layer, with no load, the averaged offset can be used at the client though sub microsecond accuracy is not guaranteed. When the hub-network is loaded the offset varies highly due to back off effects added to the processing delays.

Positive and Negative Offsets

Figure 3, gives details on the significance of positive and negative offsets calculated at the client. If offset various between positive and negative values, then the directional delays between client-server and server-client can be assumed to be symmetric. In some of the data recorded below one notices, that the offset is either always negative or positive, which arises because of asymmetrical delays in either direction.

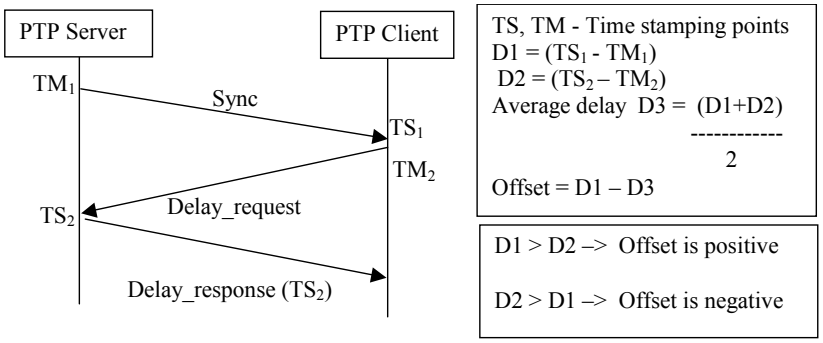


Figure 3: Significance of Positive and Negative Offsets

SWITCHED 802.3 ETHERNET

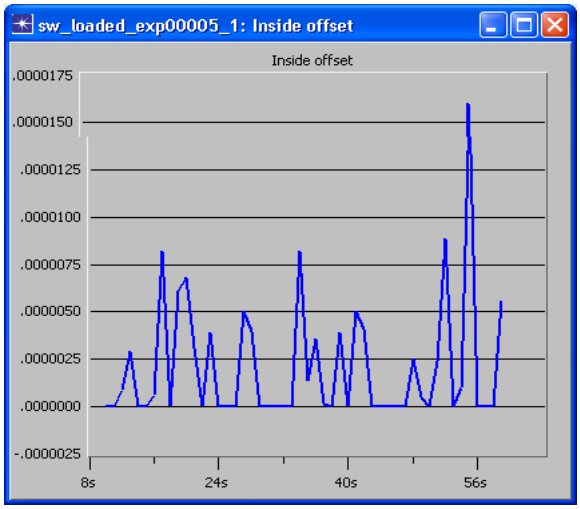
The Opnet models are similar to hub-based Ethernet, with the hub replaced by a switch.

NO LOAD SWITCHED NETWORK

Time stamping inside the MAC layer: The network delay in this case was calculated to be a constant at 13.693 µs and the averaged offset at the client was constant at 0. After estimating the transmission and propagation delays, the switch delay was calculated to be 2.0797 µs. Experiments conducted on a switched 802.3 Ethernet test-bed at Spectracom gave the switch delay to be 15.28 µs. This variability is acceptable as switch models vary and the delays are not expected to be constant. The switch model used in Opnet simulations was a non-blocking switch with 'infinite' processing capacity, infinite input and output buffers.
Time stamping outside the MAC layer: The average offset calculated was 224.71 µs, and the variance was 1.173 ms^2. For other seed values, the average offset was recorded as -2.5 µs, and -31 µs.
Inference: As noted earlier for the hub-based networks, if time-stamping is done at the PTP layer, then it is best to use the averaged offset at the client. However the offset is not within sub-microsecond range.

LOADED SWITCHED NETWORK

Time stamping inside the MAC layer: Load on the switch affects the synchronization accuracy achieved even with time stamping inside MAC layer. Table 1 shows effect of load on offset at PTP client. As the load increases the offset increases and is beyond sub-microsecond range. The reasons for this can be attributed to packets stored in the output buffers, which is rate-limited by transmission rate of the media.
Time stamping outside the MAC layer: The data in table 2, was obtained with time-stamping at the PTP layer. The delays in this case are highly variable and do not show dependency on traffic or



Offset with 35% link utilization (seed 99)

load in the network, because the processing delays are predominant.

Table 1: Time stamping inside the MAC layer

Load	3.5%		17.5%		35%		60%	
Seed	99	1177	99	1177	99	1177	99	1177
Offset in µs	0.015	0.379	1.02	1.3	2.04	2.42	4.26	5.67

Table 2: Time stamping outside the MAC layer

Load	3.5%		17.5%		35%		60%	
Seed	99	1177	99	1177	99	1177	99	1177
Offset in µs	-110	-16.7	4.9	79.7	17	-36.5	216	-7.8

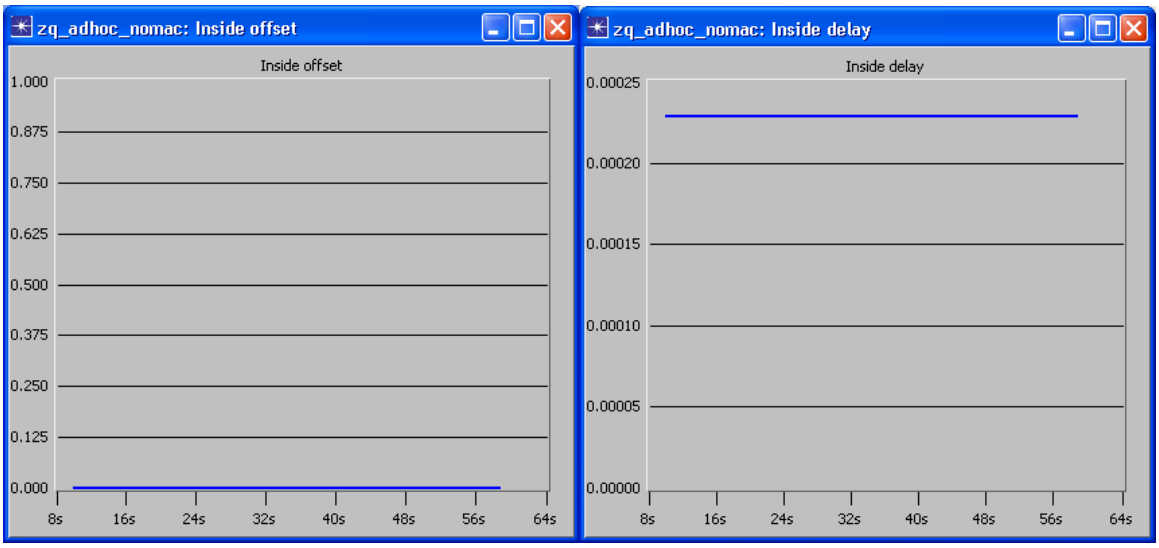
Inference: With time-stamping inside MAC layer, and using non-blocking switches, delays due to packet buffering in switch affects synchronization accuracy as traffic (especially at client port) in the switched network increases. At around 35% load this effect is around a couple of microseconds, which may be acceptable in some cases. The offset is positive because the port at which the PTP server is connected handles only the traffic to/ from the server, whereas the client handles other traffic beside PTP traffic.

802.11B AD HOC WIRELESS NETWORKS

NO LOAD AD HOC NETWORK

In this scenario, we had only the PTP server and client communicating with each other in ad hoc mode.

Time Stamping inside the MAC layer: In this case the offset at client was constant at 0 and network delay was constant at 229 µs. Though 802.11-based devices back-off, as we are performing time-stamping at the physical layer, back-off does not influence either the delay estimated in the network, or the offset.

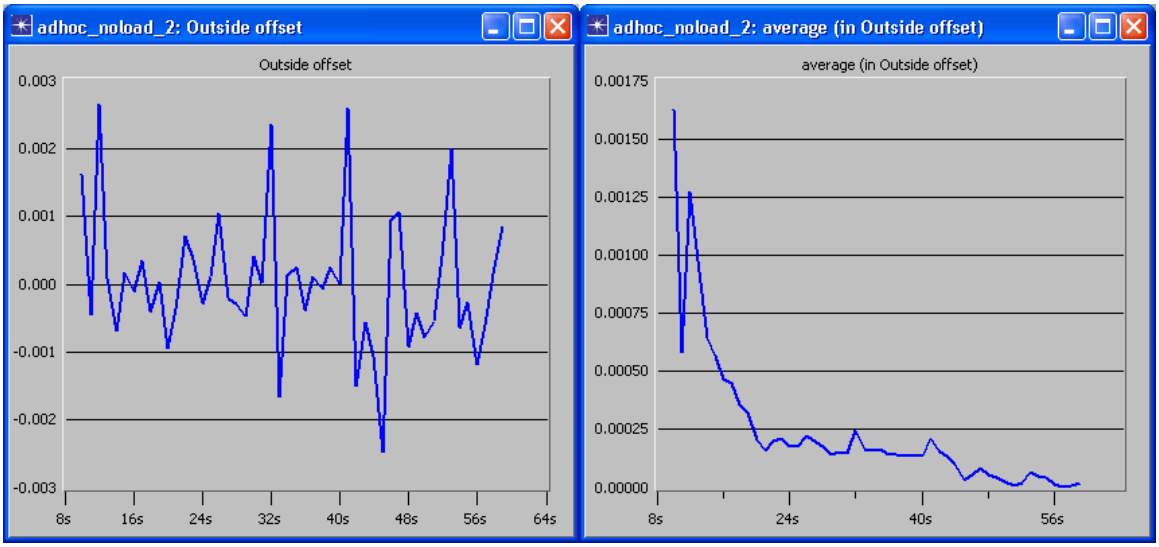


Offset Network delay

With a frame size of 2516 bits, and transmission speeds of 11 Mbps, the transmission delay is 228.72 µs, propagation delay is 0.33 µs hence the network delay is 229.05 µs, which is close to the value noticed in the graph.

Time stamping inside the MAC layer - with back-off delays: In this case, as we intended to study the back-off effects, we did not introduce the M/G/1 queue process model for the protocol processing delays. Time-stamping was done when the frame first arrived in the transmit buffer. Back-off or any retransmissions would introduce variable delays. The average offset measured varied from -325 µs, -310 µs, and -350 µs for different seed values.

Time stamping outside the MAC layer: The average offset recorded for different seed values was 35 µs, -100 µs and 22.96 µs respectively i.e. it varied between positive and negative values.
Inference: The no load ad hoc test scenario is not a realistic scenario, but this experiment gives us an idea of the minimum delays to be expected between a PTP server and client.
In the test case where we perform time-stamping inside the MAC layer but with back-off we are able to assess the variable transmission delays due to back-off only. The high negative offset recorded indicates that when the server transmits – it is experiencing an idle media, as it sends a PTP packet once every 1 second. However, the client responds with a 'delay_request' message as soon as it received the PTP 'sync' packet, hence as it has just experienced a non-idle media, it backs off. This results in an asymmetric network delay experienced between the server and the client. The variability is due to the random number picked by the client for back-off. This experiment gives insight into a situation that should be avoided in wireless networks i.e. asymmetric delays – which could affect the offset estimated at the client considerably.



Offset (seed 1177) Avg. Offset

The third case provides another interesting observation, where including the MAC layer delays has reduced the highly negative offset obtained in the previous case. This is because, when the client is ready to transmit after the protocol processing delays, it may also encounter an idle media.

Though averaging the offset over successive values reduces the offset error, the value obtained is still unacceptable.

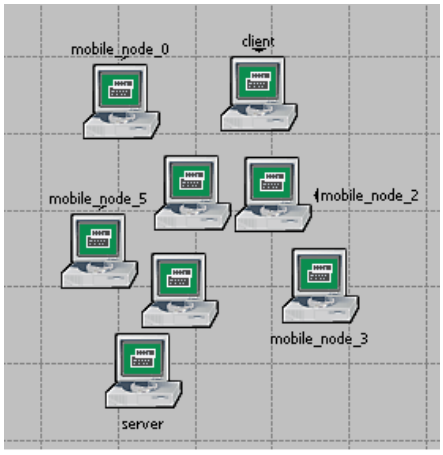


Figure 4 Loaded Ad hoc Network

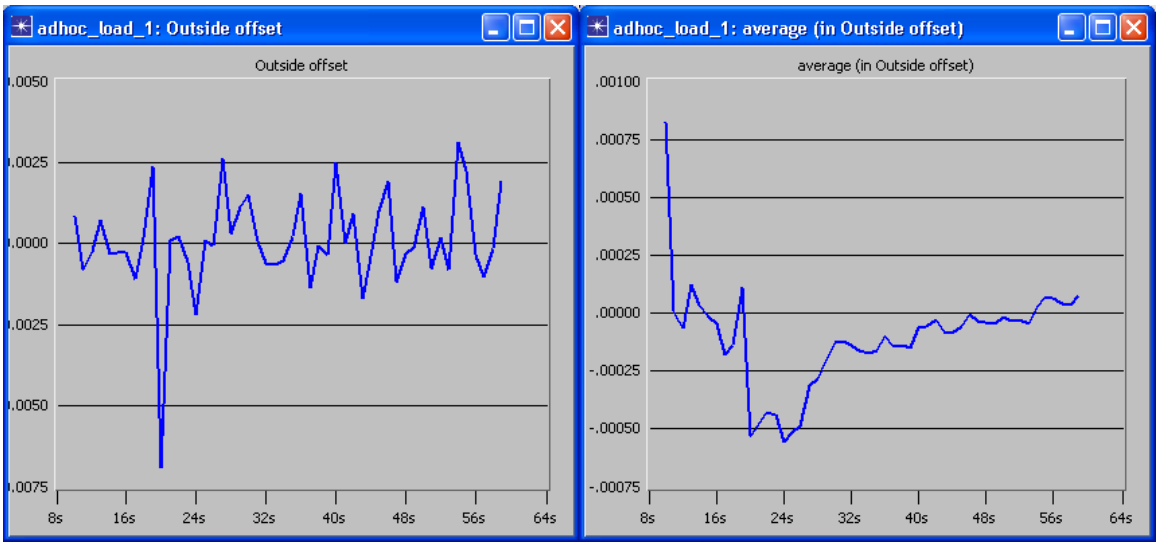
LOADED AD HOC NETWORK

The ad hoc network was loaded to 10%.

Time Stamping inside the MAC layer: The performance observed in this case was similar to that observed in the no-load ad hoc network case as we time-stamp when the PTP packets leave the physical layer or as soon as they are received at the receive buffer.

Time stamping inside the MAC layer – with back-off delays: The averaged offset recorded for different seed values were –290 µs, -210 µs, and -285 µs.

Time stamping outside the MAC layer: The averaged offset recorded for various seed values were 130 µs, 73.76 µs and 12.27 µs.



Offset (seed 99) Avg. Offset

Inference: Noticeable here is case 2 where we recorded a less negative offset as compared to the previous no-load case. This is because the server at times sees a busy medium, because there is traffic in the network. Introducing the protocol processing delays reduces the asymmetric delays experienced between the server and client.

802.11B INFRASTRUCTURE NETWORK WITH WIRED SERVER

In these scenarios we assumed that the there is an Access Point (AP) and all communications are via the AP. The PTP server however is wired to the AP, but the client is communicating with AP over the wireless media.

NO LOAD INFRASTRUCTURE NETWORK

In this case, the PTP server and client are the only devices communicating with each other via the AP.

Time Stamping inside the MAC layer: The offset recorded was constant at 0. This is because the AP does not back-off as it sees an idle media when it forwards the 'sync' message from the PTP server. The client, is time-stamping the PTP packets as they leaves the physical layer and recording the time stamp of a PTP packet as soon as it arrives at the receive buffer.

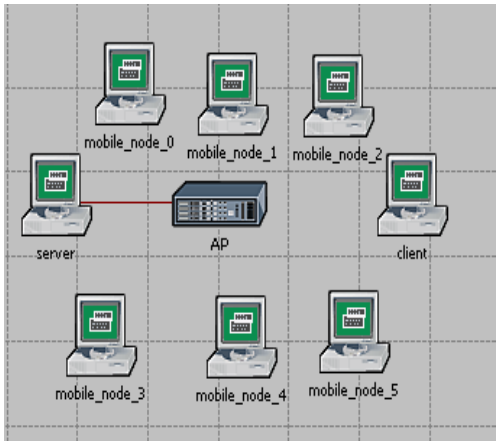


Figure 5 Loaded Ad hoc Network - Wired Server

Time stamping inside the MAC layer – with back-off delays: The average offset recorded for various seed values was –335 µs, -343 µs, and -360 µs.

Time stamping outside the MAC layer: The average offset recorded for various seed values was 20 µs, -12 µs, and 160 µs.

Inference: The effect here is similar to the unloaded ad hoc network case, as the AP always sees an idle media, whereas the client always sees a non-idle media. Introducing processing delays at MAC layer especially at the client reduces asymmetry observed in network delays by the devices.

LOADED INFRASTRUCTURE NETWORK

The network was loaded to 10%.

Time stamping inside the MAC layer: The average offset was recorded as 6 milliseconds.

Time stamping inside the MAC layer – with back-off delays: The average offset recorded was 4.75 milliseconds.

Time stamping outside the MAC layer: The average offset is around 5.25 milliseconds.

Inference: The offset noted in the three cases is very much higher than the previous cases. The reason for this is because of queuing and back-off at the AP due to load in the network. In the first case, even though we timestamp at the client just as the PTP frame is leaving or is received at the physical layer, and the frames from the client to the server make it faster. However frames sent by the server arrive at the AP, which is backing off and has queued packets and delays the packets, hence the offset is highly positive.

In case 2 as the client is also backing off, the highly positive offset is slightly reduced, the effect of the packets queued on the wireless side of the AP however still introduces asymmetric delays.

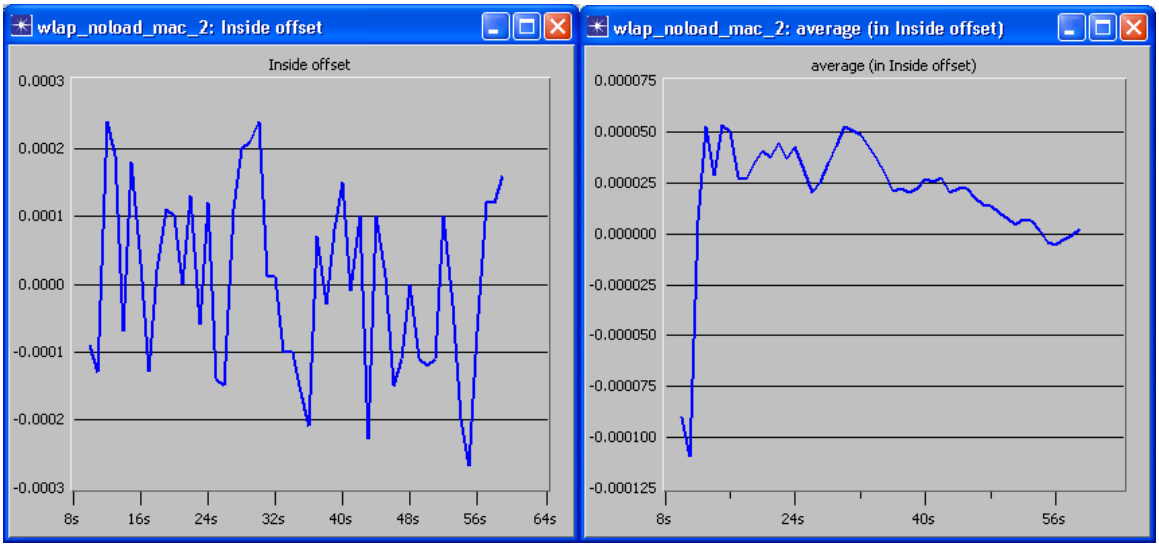
In case 3, protocol-processing delays introduced at client and server, null each other and the client at times encounters an idle media after processing delay, hence the higher positive offset as compared to case 2.

802.11B INFRASTRUCTURE NETWORK - WIRELESS SERVER

In these scenarios we assumed that the PTP client and server communicate wireless via AP.

NO LOAD INFRASTRUCTURE NETWORK

In this case, the PTP server and client are the only devices communicating via the AP.



Offset (seed 1177) Avg. Offset

Time Stamping inside the MAC layer: Average offset recorded for various seed values were 35 µs, 13.2 µs, and 2.6 µs.

Time stamping inside the MAC layer – with backoff delays: Average offset recorded for various seed values were –295 µs, -305 µs, and -315 µs.

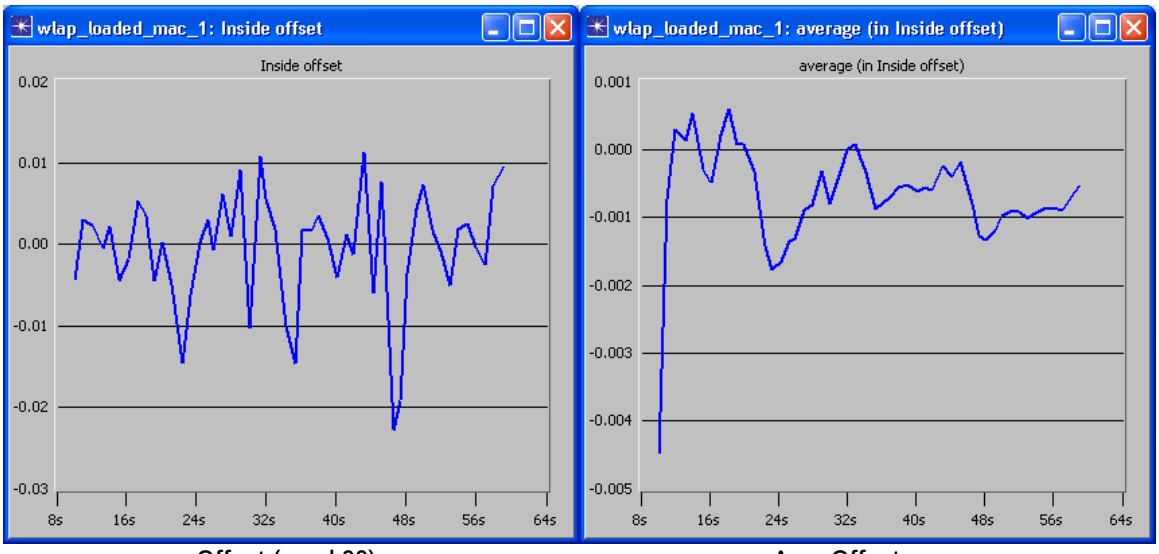
Time stamping outside the MAC layer: Average offset recorded for various seed values were –156 µs, -180 µs, -21.59 µs.

Inference: In case 1 the time-stamping at client and server is done at the physical layer, the AP encounters a busy media when it forwards the frames from server or client. However the simulations conducted here, shows positive offset always, due to random seed effect.

In case 2 the AP backs off in either direction i.e. while sending to client and to server and the net effect can be considered as nulled. The client always backs off because it tries to send as soon as it receives PTP packets, whereas the server, encounters mostly an idle medium. Hence there is a high negative offset.
In the case 3 the client may not back off at times because of processing delays, hence the effect is reduced.

LOADED INFRASTRUCTURE NETWORK

The network was loaded at 10%.



Offset (seed 99) Avg. Offset

Time Stamping inside the MAC layer: The average offset for various seed values was recorded as 675, 900, -520, 552 µs.
Time stamping inside the MAC layer – with back-off delays: The average offset for various seed values was recorded as 400 µs, -62 µs, and -620 µs.
Time stamping outside the MAC layer: The average offset for various seed values was recorded as 476, -180, 437 µs.
Inference: In case 1 though time stamping is done at physical layer, the AP is facing a busy media and there are queued packets in either direction due to load in the network. Hence offset is varying between positive and negative values.
In case 2, the server and client are backing off, due to the loaded network. The server may at times encounter an idle media as it sends only once in 1 second, where as the client always encounters a busy media.
In case 3, symmetrical delay effects are experienced by the traffic in either direction.

CONCLUSIONS

From the results recorded and discussed above, it is very clear that it is difficult to use 1588 PTP in a switched Ethernet especially if it is loaded beyond 35%. However the introduction of transparent clock at the switch may solve this problem. The performance in wireless networks is highly variable depending on the configuration and load of the network. These studies gave us an insight to the cause of these variations. This will be helpful in identifying a solution to overcome the loss in synchronization in wireless networks between the PTP client and server.

A SIMULATION STUDY - PERFORMANCE OF IEEE 1588 OVER ETHERNET AND IEEE 802.11B WLANs

Nirmala Shenoy, PhD – Rochester Institute of Technology

John Fischer, Paul Myers, Zeping Qui - Spectracom

Rochester, New York USA October, 2005

This research was funded in part by CEIS, NYSTAR-designated Center for Advanced Technology

Scope

- Introduction
- Opnet Simulations
  - 802.3 Ethernet, 802.11b WLAN
- PTP - Time Stamping Points
- Test Cases
  - Observed Data
  - Inferences
- Conclusions

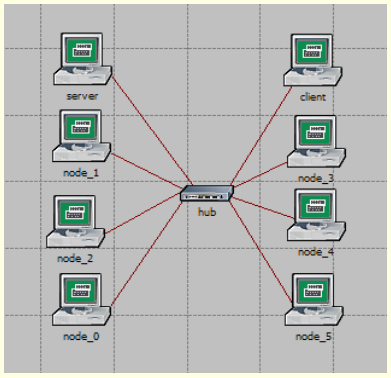
R·I·T SPECTRACOM

Introduction- the study

- PTP clock synchronization accuracy
  - 802.3 Ethernet
  - 802.11b WLAN
- Effects on clock accuracy
  - Network configuration
  - Type of devices
  - Time stamping points
  - Varying loads
- Inference, Pointers to solutions

R·I·T SPECTRACOM

Opnet Model



- Hub – 802.3 Ethernet
- 100 Mbps
- No - Load network
- Loaded Network
  - More devices and traffic
- PTP server, client

R·I·T SPECTRACOM

Time Stamping Points

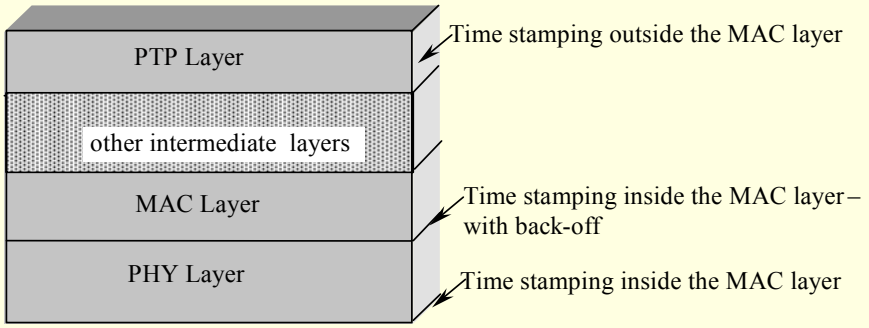


Figure 1 Time stamping

R·I·T SPECTRACOM

Protocol Processing Delays

- M/G/1 queue process model
- Lumps all protocols in one queue

The M/G/1 queue model is defined by the equation
Γ = 1/μ + ω , Γ is the total system waiting time,
μ is service rate, ω is queue waiting time.
ω = η(1/μ^2 + σ^2)/2(1 – η/μ), η is arrival rate for jobs at the queue, σ^2 is the variance.
For
σ^2 = .0001 ms^2, η = .6 to.95 jobs/ms, μ = 1 job/ms
Γ(maximum) = 10.5 ms and Γ(minimum) = 1.75 ms

Hub based 802.3 Ethernet

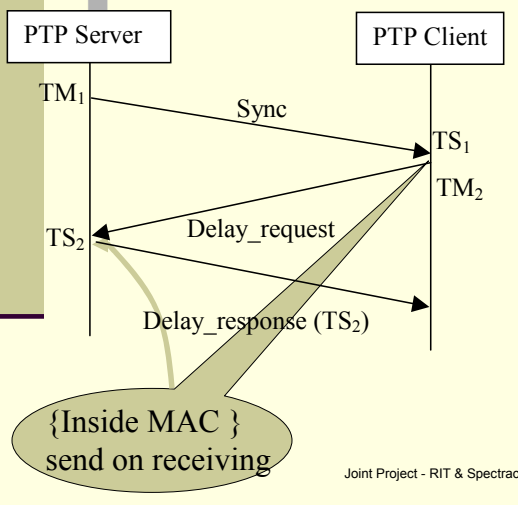
- No Load
- Loaded

	Offset	Variance
Inside MAC	0	-
Outside MAC	79.1258 μsec	1.272 ms [2]

Load %	4.5	9	20	45	70	70
Offset in μsec	41	18	364	397	-100	-650

Inside MAC – Offset is 0

Offset Measurement



$D1 = (TS_1 - TM_1)$
$D2 = (TS_2 - TM_2)$

Average delay $D3 = \frac{(D1+D2)}{2}$

Offset = D1 – D3

D1 > D2 –> Offset is positive

D2 > D1 –> Offset is negative

R·I·T SPECTRACOM

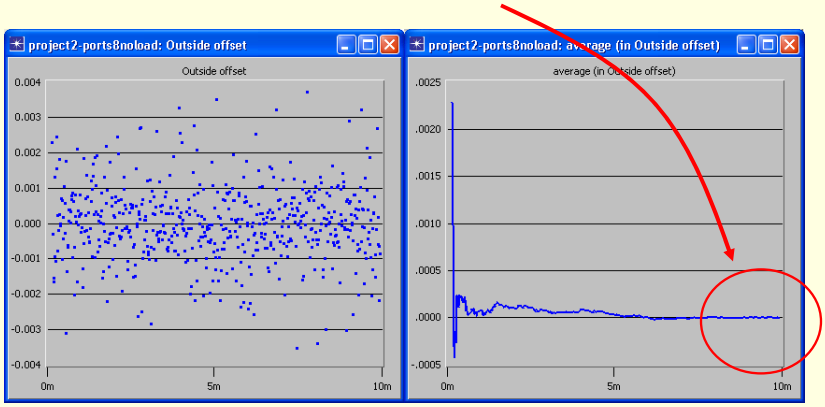
Positive and Negative Offset

- Negative Offset
  - Delay measured from client to server is higher
- Positive Offset
  - Delay measured from server to client is higher
- Offset is highly variable
  - Not in sub-microsecond range
  - Broadcast media
    - Collisions, back off
  - Processing delays

R·I·T SPECTRACOM

Hub based 802.3 Ethernet

- Offset averaging effects



R·I·T SPECTRACOM

Switched 802.3 Ethernet

- PTP server – on a dedicated port
- No Load

	Offset in μsec	Variance
Inside MAC	0	-
Outside MAC	224,-25,-3.1	1.173 ms [2]

Varying seeds

R·I·T SPECTRACOM

Switched Ethernet

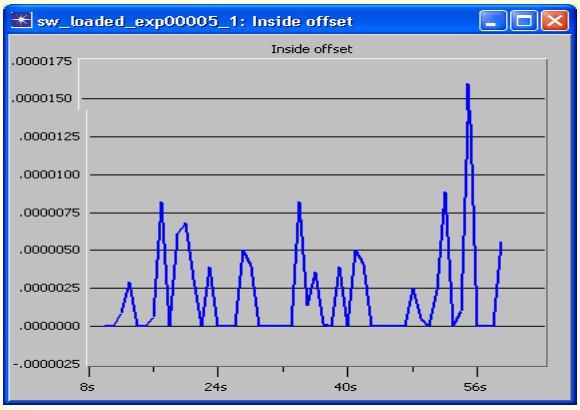
- Offset in μsecs, Load %

Acceptable?

Inside MAC				
Load	3.5	17.5	35	60
Offset	0.015, 0.379	1.02, 1.3	2.04, 2.42	4.26, 5.67
Outside MAC				
Load	3.5	17.5	35	60
Offset	-110, -16.7	4.9, 79.7	17, 36.5	216, -7.8

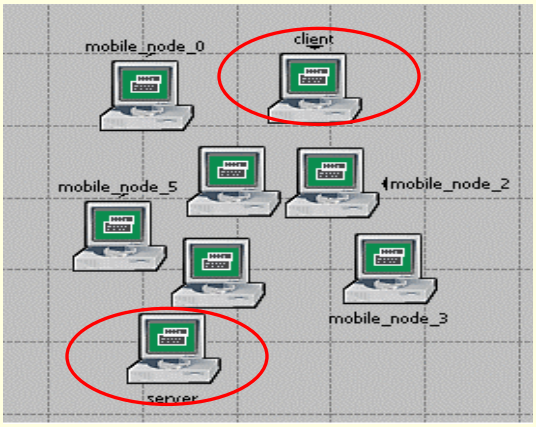
Switched Ethernet

- Asymmetric Delays – 35% load



WLAN – Ad Hoc

- Opnet Model



WLAN – Ad Hoc

- No load – Only PTP Client and Server
- Assumption on sync, delay_req transmission

Time stamping	Offset in μsec
Inside MAC	0
Inside MAC- backoff	-325, -310, -350
Outside MAC	35, -100,22.96

Client sees a busy media always

WLAN – Ad Hoc

- Loaded vs no load – Offset in μsecs

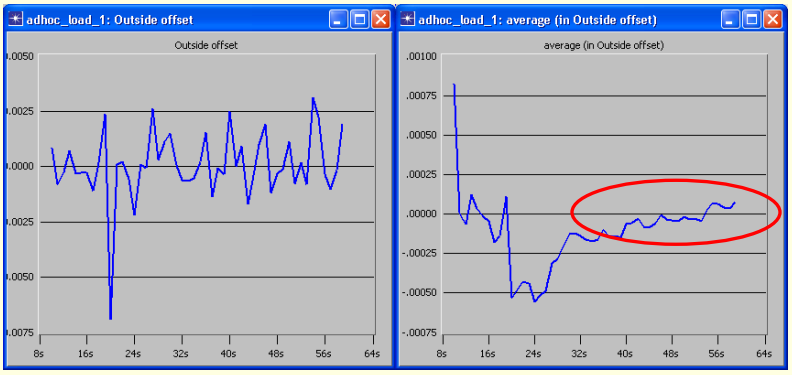
No Load	
Inside MAC – back off	-325, -310, -350
Outside MAC	35, -100,22.96
Load – 10 %	
Inside MAC – back off	-290, -210, -285
Outside MAC	130, 73.76, 12.27

Client sees a busy media always

Server may see a busy media

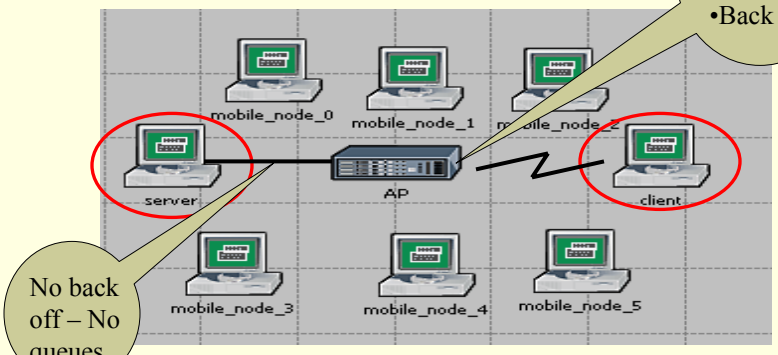
WLAN – Ad Hoc

- 10% load – Outside Offset



WLAN - Infrastructure

- Wired Server



•Packets queued

•Back off

No back off – No queues

R·I·T SPECTRACOM

WLAN - Infrastructure

- Wired Server – No Load – similar to ad hoc
- Wired Server – Loaded 10%

AP backs off Packets queued

Time stamping	Offset in msec
Inside MAC	6
Inside MAC- backoff	4.75
Outside MAC	5.25

Client also backs off

Processing delay reduces client back off effect

R·I·T SPECTRACOM

WLAN – Infrastructure (wireless)

NO LOAD	
Inside MAC	35, 13.2, 2.6
Inside MAC – back off	-295, -305, -315
Outside MAC	-156, -180, -21.59
LOADED 10%	
Inside MAC	675, 900, -520
Inside MAC – back off	400, -62, -620
Outside MAC	476, 180, 437

Symmetric effects

R·I·T SPECTRACOM

Conclusions – Wired Ethernet

- Poor (msec) performance if time tagging is at the App Layer
  - Averaging may help
- Good (1 usec) performance through switch on lightly loaded (<25%) ports
  - <10 usec through switch with heavy loads
  - Transparent Clocks will correct this

R·I·T SPECTRACOM

Conclusions - Wireless

- Asymmetry in the Client => Server vs. the Server => Client path is the biggest error source. It prevents the Delay_Req measurement from compensating correctly.
- Can this be overcome?
  - Unlink Dely_Req from Sync transmissions – (no load)
  - Access Point has to be PTP sensitive
    - Preferably with 'Inside MAC' time stamping

A Security Analysis of the IEEE 1588 Standard

Jeanette Tsang & Konstantin Beznosov

October 12, 2005

Laboratory Education and Research in Secure System Engineering (LERSSE)
University of British Columbia

What will happen if...

Photo from:
:http://www.edwards.af.mil/articles98/docs_html/splash/jan98/cover/page_5.html

And what if...

Photo from: www.aandbfoundry.com/ products.html

Outline

1. Objectives
2. Assumptions
3. Discussion of possible attacks
4. Results summary
5. Conclusion

How do you know PTP is “secure”?

- No security analysis has been done

- Confidentiality
- Integrity
- Availability

Our Objectives

1. Identify PTP security vulnerabilities for generic attacks
2. Identify PTP-specific vulnerabilities
3. Suggest countermeasures

Assumptions

1. Closed network
   - i.e., no direct or indirect connections with the Internet
2. Insiders can mount active attacks
   - i.e., remove, modify, and inject messages
3. No IP-level data protection
   - e.g., IPSec

Attacks

Attacks Identified

1. Modification
2. Masquerading
3. Delay
4. Replay
5. Denial of service

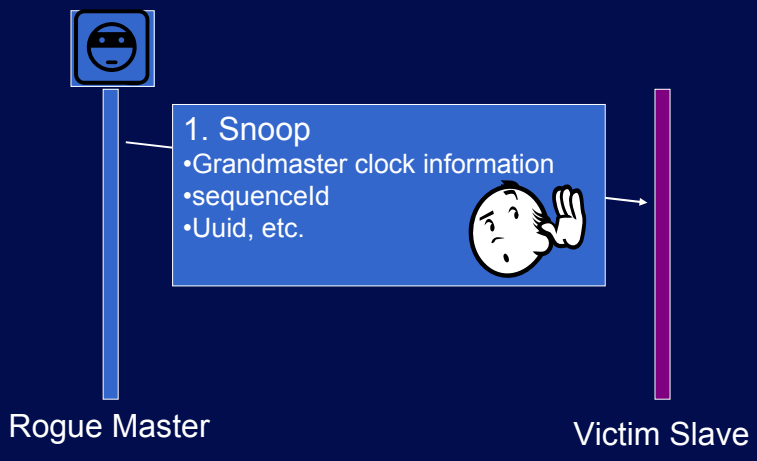
Attack I: How to Masquerade as the Master Clock

Two ways:

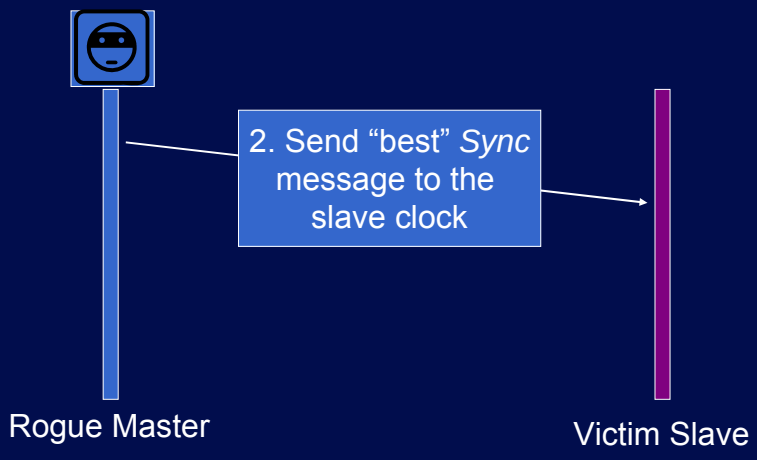
1) Impersonate Current Master Clock
   - "Steal" current master clock identity

2) Switch the slave clock to the rogue master clock
   - Win the Best Master Clock (BMC) election

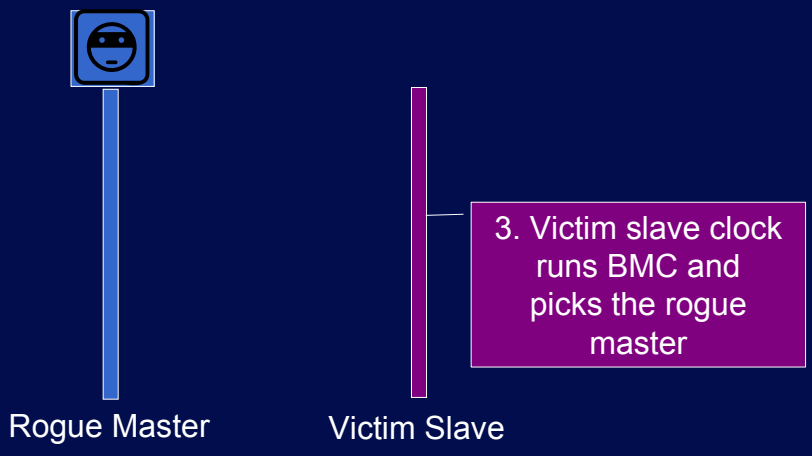
How to Win BMC Election (1/4)



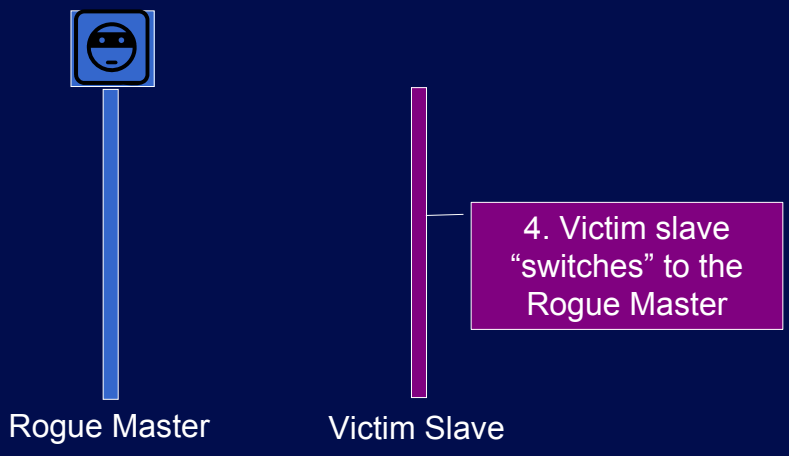
How to Win BMC Election (2/4)



How to Win BMC Election (3/4)



How to Win BMC Election (4/4)

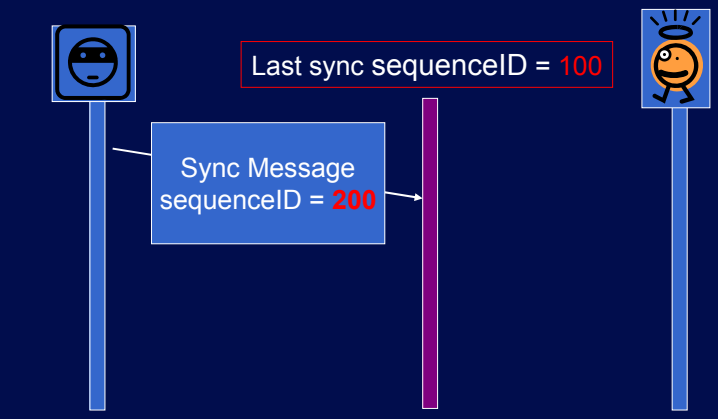


Attack 2: Depriving slave from synchronization

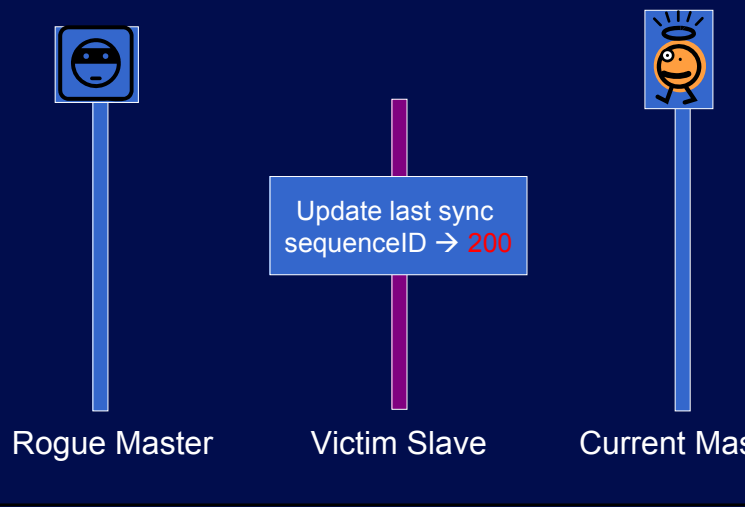
Ways to attack:

1. Block *sync* messages
   - Congestion
   - Removal
2. Make victim slave to discard good sync messages
   - *Sync* message modification
   - Illegal update of *sequenceId*

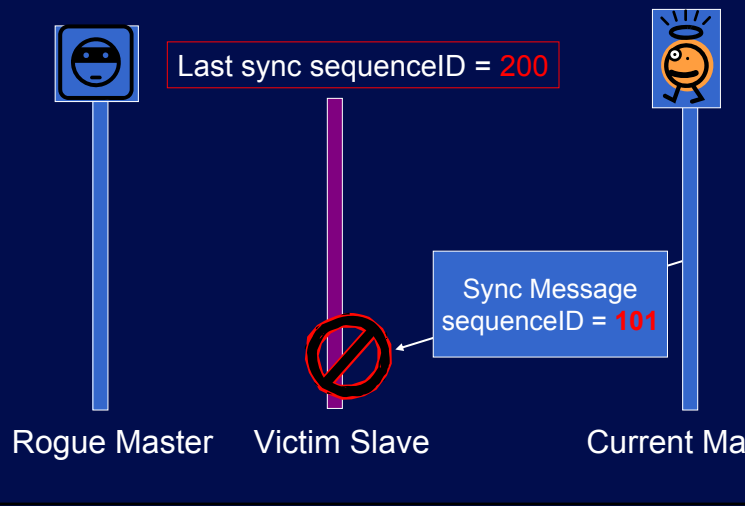
Attack 2: Illegal update of sequenceId (1/4)



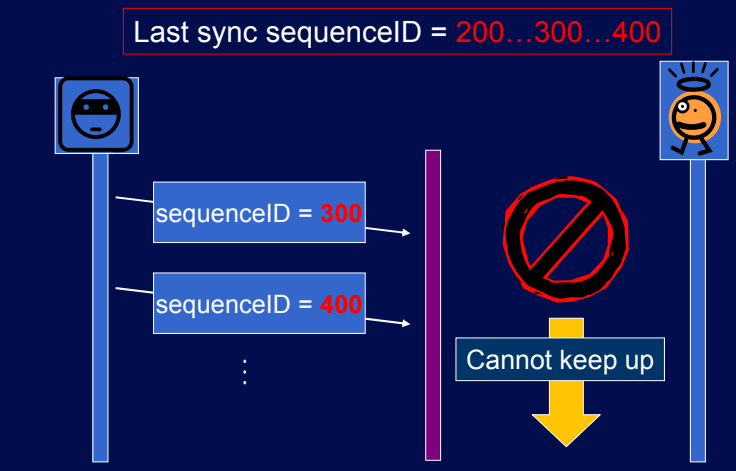
Attack II: Illegal update of sequenceId (2/4)



Attack 2: Illegal update of sequenceId (3/4)



Attack 2: Illegal update of sequenceId (4/4)



Results Summary

Attack	Effects	Countermeasures	IPsec?
Modification	•Denial of Service •Incorrect resynchronization •Changing clock hierarchy	•Cryptographic integrity protection	Yes
Masquerading	•resynchronization	•Centralized or chained authentication mechanism	Yes
Delay	•Delay in timing messages •Timeout of synchronization process •Increase in offset calculation	•Algorithm to detect abnormal timestamp •Back up plan using previous timing records	No

Results Summary

Attack	Effects	Countermeasures	IPsec?
Replay	•Disturbance of message sequence •Saturate process queue •Congest network paths	•Authentication mechanism •Tunneled connection	Yes
Denial of Service	•Small-scaled: Affect accuracy of synchronization •Big-scaled: Put halt on the whole PTP system	•Physical protection •Pay precautions to other malicious attacks •Monitor traffic	No

Conclusions

- Presented two attacks:
  - Masquerading
  - Depriving slave from synchronization
- Countermeasures:
  - Integrity protection
  - Authentication mechanism
  - Tunnelled connection
  - Monitor network traffic
  - Detect abnormal timestamp

More information
lersse.ece.ubc.ca

IEEE1588 Security Extensions: Requirements and proposed solutions

IEEE1588 Conference 2005 (v2)

Author: Stephan Schüler, Siemens Communications

Schueler.stephan@siemens.com

SIEMENS

Agenda

- Threats
- Countermeasures
- Implementation: Authentication and integrity protection, Message extensions
- Key Management

Threats on PTP systems

Eavesdropping

The attacker has access to the LAN and uses sniffing tools that enables him to eavesdrop and analyze PTP-Protocol (traffic analysis).
This passive attack is a precondition for further active attacks.

Man in the Middle attack

e.g. If a switch / transparent clock modifies a message with the goal of worse the accuracy

Replay attacks

Replaying eavesdropped / recorded PTP-Messages to disturb the PTP-Ports

Denial of Service attack

The attacker sends a flood of messages to overload the PTP-Ports

Masquerading

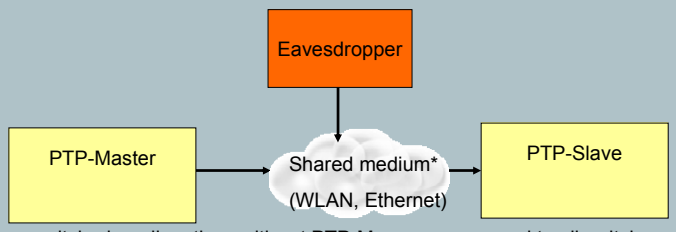
A hacker uses the identity of an other user to remain undetected.
Misuse of Service through Unauthorized access to management I/F of PTP-Ports,
e.g. change PTP-Parameters with PTP_MM_SET_.

Misuse of Service

through Unauthorized access to management I/F of PTP-Ports, e.g. change PTP-Parameters with PTP_MM_SET_.
through Unauthorized participating in PTP subdomain, e.g. taking over the PTP-Master role and send invalid time information.

Countermeasures to Eavesdropping

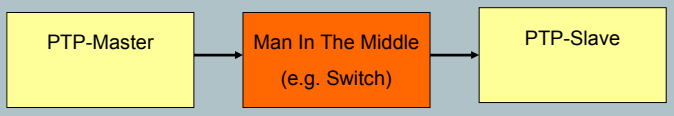
- It is not required to have confidentiality for PTP-Traffic
  (timestamps are no sensitive data)
- Encryption would be the measure, but this makes the timestamping complicated
- Today there are no application areas requiring confidentiality for PTP-Traffic



*) Even in a switched medium the multicast PTP-Messages are send to all switch-ports

Countermeasures to Man in the Middle attacks

- Can't be prevented in the PTP-Port, but:
- Checking the plausibility of the timestamps:
  filter algorithms in the PTP slave detecting timestamps
  out of range to improve the robustness

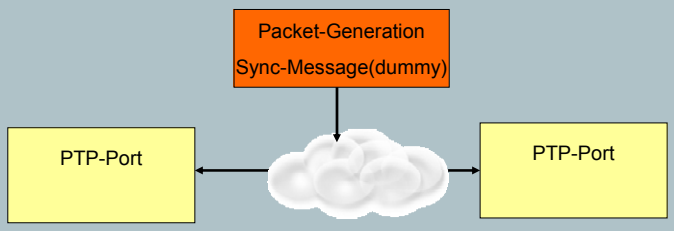


Countermeasures to Replay attacks

- Replay attacks can be identified when the same sequence number occurs twice
- Using sequence number field in the messages.
- Sequence numbers are already included in the header of all PTP-Messages and need not to be included in a security extension
- Checking the plausibility of the timestamps (filter algorithms in the PTP slave)

Countermeasures to DoS-Attacks

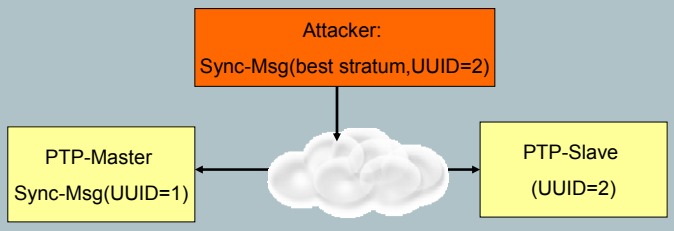
- Can't be prevented in the PTP-Port, but:
- Send management alarm message (Alarm: "message discarded"), if DoS attack was detected (e.g. flood of packets)
- May be a new PTP management message or in case of SNMP a trap



SIEMENS

Countermeasures to Masquerading

- Precondition: Attacker has eavesdropped a message from the PTP-Port with UUID=2
- Attacker takes the identity of the PTP-Slave with UUID=2
- Attacker may become PTP-Master



UUID= universal unique Identifier

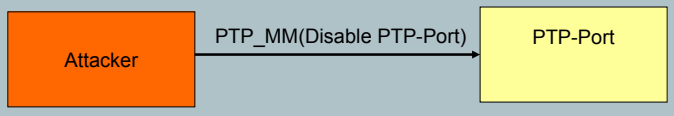
SIEMENS

Countermeasures to Masquerading

- **Authentication** can prevent masquerading
- Option 1: Group authentication of PTP-Ports
  - every PTP-Port gets the same cryptographic key (the „shared secret“)
  - The receiver can verify, that the message comes from one in the group
- Option 2: Key management system
  - centralized authentication
  - A central group controller (GC) is required
  - GC can be located in the grandmaster clock
  - (Details described later)

Countermeasures to Misuse of service (by User)

- Authenticate the user who wants access to the management I/F
  - Authentication of PTP_MM messages using
    encrypted hashed message authentication code (e.g. HMAC-SHA1)
  - Use other secure management interface like
    - SNMPv3 (requires a **MIB / private MIB** for PTP)
    - secure Web-I/F with ssl (https)



Countermeasures to Misuse of service (by Device)

- Encrypted Message Authentication Code (e.g. HMAC-SHA1)
- Group authentication (one cryptographic key for one subdomain)
- Checking the integrity of the message
  - recalculating the appended integrity check value (e.g. with HMAC-SHA1)
- Alternatives (if available):
- Configure separate VLANs for PTP ports
  - the switches and devices must be VLAN-aware
- Use IEEE802.1X for authentication
  - requires authentication server, e.g. RADIUS-Server in the network

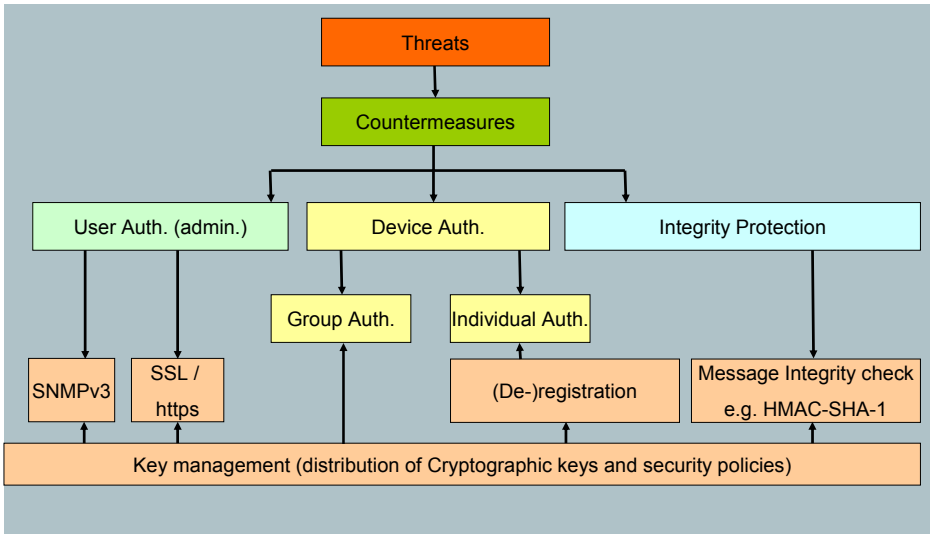
SIEMENS

Security for PTP (Overview 1)

- Defining message extension fields for security
- Defining cryptographic algorithms and keys to use
- Group authentication of PTP-ports (based on PTP subdomains)
- Secure Administration of PTP-Ports
- Protection against “replay attacks”
- Guidelines:
  - reuse suitable existing security concepts
  - Use standardized algorithms

SIEMENS

Security for PTP (Overview 2)



SIEMENS

Integrity protection and group authentication

- Integrity protection and authentication between the PTP-Ports
- A one way hash value is calculated over the PTP-Message (payload)
- The hash value is encrypted with the shared secret
- Use HMAC-SHA-1-96

SIEMENS

HMAC

- IETF RFC 2104 (1997), *HMAC: Keyed-Hashing for Message Authentication*
- HMAC is used together with an hash algorithm, e.g. SHA1

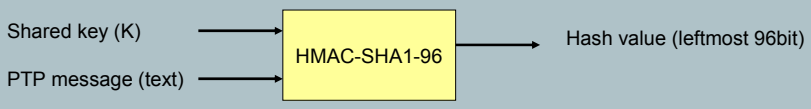
- **Example: HMAC-SHA1-96**

is the truncated 96-bit cryptographic hash value of the 160-bit SHA1 computation.

The 96 leftmost bits of the network byte order representation of the hash value

shall be used as the result.

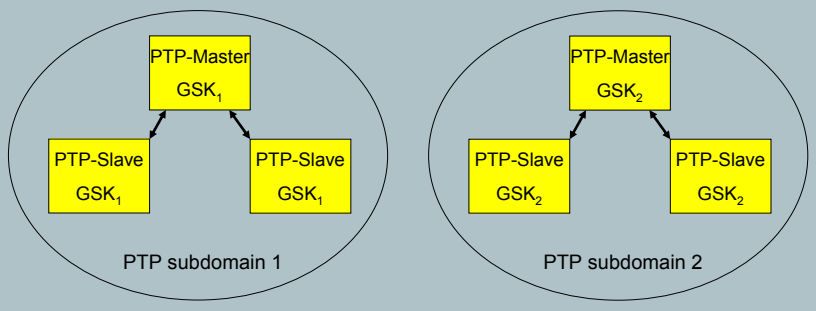
RFC 2104 describes the procedure with the **secret key *K*** set to the ***shared secret***
(20 byte SHA1-hashed password) and ***text*** set to the ***PTP message***.



SIEMENS

Keys: Shared Secret

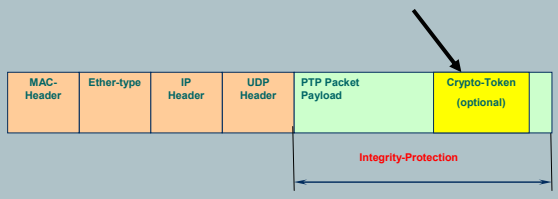
- The shared secret is a common Group shared key (GSK=K) for all PTP-Ports of one PTP subdomain
- A symmetric key is used for fast calculation
- Only one key K for the whole PTP subdomain
- The key is stored in every PTP-Port as a 20 byte SHA1-hashed password



SIEMENS

Message Extension: Crypto Token

structure of a PTP packet with the new security extension (Crypto Token):



- Crypto Token can be included in every PTP-Message
- Crypto Token is optional
- It carries the keyed hash value, which is calculated over the complete PTP packet payload
- (without UDP/IP-header) and additional information about the used algorithm
- Before calculation the Crypto Token-Fields will be set to zero

Elements in the Crypto Token

Algorithm
Keyed hash

Algorithm to be used with HMAC:

0= SHA-1 (NIST FIPS PUB 180-1)

1= SHA-256 (NIST FIPS PUB 180-2)

Keyed hash value calculated over the complete PTP packet payload (without UDP/IP-header)

truncated to the leftmost n byte

(HMAC-<algorithm>-<n*8>),

n=12. Default for long messages only, e.g. HMAC-SHA1-96*

n=2. Default for short messages, e.g. HMAC-SHA1-16*

*) discussed today, but truncated hash should **not be < n=10 byte** (collision problem: the attacker can make an offline analysis and find a key which produces a hash with the same leftmost n bits.

HMAC:

- IETF RFC 2104 (1997), *HMAC: Keyed-Hashing for Message Authentication*
- The message will be hashed with a hash-algorithm, e.g. SHA1
- The hash value will be encrypted with the shared secret

Key Management: Purpose

First approach (as described before):

- We establish a Group Security Association (GSA), using a group key (GSK)
- We use a symmetric GSK (to achieve low computational workload, …)
- We configure the GSK and the security policies (SP) separately and manually in each GM

SP = manually configured parameters like algorithm to use, Key length, …

Second approach (adding the key management):

- We introduce a key management for automatic key distribution (and rekeying) of the group session key (GSK)
- Requirements:
  - a Group Controller / Key server (GC/KS)
  - establish a secure tunnel between the GC/KS and the GM for the key distribution
- The secure tunnel can be established in two ways:
  - using symmetric keys (the Master Key, MK)
  - using asymmetric keys (public and private keys)

SIEMENS

Key Management: Group controller model

General Requirement for PTP key management:

The proposed group key management architecture to be used for PTP is based upon a G**roup Controller Model, described in:**

- RFC 2093 Group Key Management Protocol (GKMP)
- RFC 2094 GKMP Architecture
- RFC 4046 MSEC Group Key Management Architecture

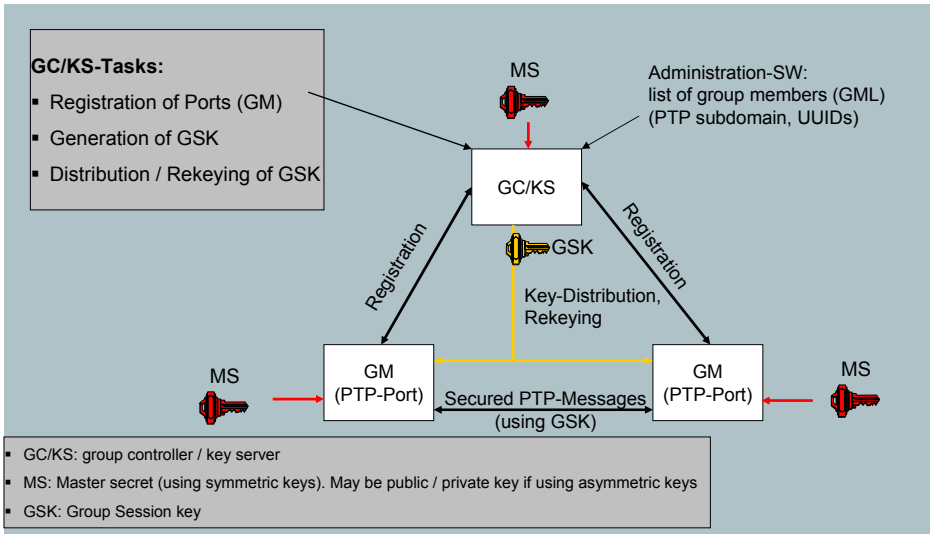
The group owner designates a group controller (GC) for member registration and rekeying. with a (**floating**) single group owner as the root-of-trust.

Special Requirement for PTP key management:

It shall be possible that the GC is floating (changing with the PTP-Master change to achieve redundancy

SIEMENS

Key Management: Group controller model



SIEMENS

Key Management: Registration Protocol

Purpose:

- Distribution of Group key
- Rekeying of group key

2 Alternatives:

1.) Pull: GM registers at GC to get the GSK (e.g. to request a valid GSK)

2.) Push: CG registers all GMs (GC requires a list of GMs which has to be configured)

SIEMENS

Key Management: MIKEY-Protocol (1)

- Published in RFC3830: MIKEY: Multimedia Internet KEYing
- Supports the Group key management architecture (GKMARCH, RFC4046)
- Can be used for **peer-to-peer** and **group** communication
- Defined in Multicast Security Working Group (MSEC WG)
- It is suitable for heterogeneous (mix of wired and wireless) networks

Key Management: MIKEY-Protocol (2)

- MIKEY is a **general purpose** key management protocol
  - It defines the basic messages, and packet blocks
  - Transport of MIKEY in SIP, RTSP defined in RFC3830
  - Transport of MIKEY in TESLA is defined (draft-ietf-msec-bootstrapping-tesla-01.t
  - Transport of MIKEY in SRTP is defined in a separate document
  - Transport of MIKEY in PTP has to be defined
- MIKEY allows also a generic use through dedicated payload types
- MIKEY defines the basic messages, and packet blocks
- MIKEY doesn‘t require a special transport protocol

Key Management: MIKEY-Protocol over PTP

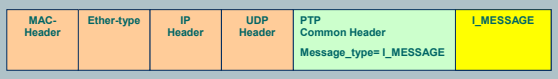
Proposed to use MIKEY-Protocol for PTP:

- Registration of PTP-Ports to GC/KS
- Distribution of GSK and security policies from GC/KS to GMs
- Rekeying of GSK

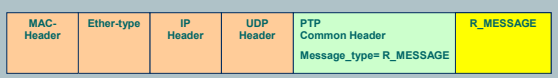
Key Management: MIKEY-Messages over PTP

- MIKEY has only two messages: Initiator Message, Responder Message
- MIKEY messages can be transported over any transport-protocol
  -> Just use the common PTP-Header for transportation

I_MESSAGE = Header, Timestamp, IDi, IDr, Security Policy, E(encr_key, {TGK}) || MAC
With encr_key = master key and TGK = session key



R_MESSAGE = HDR, Timestamp, IDr, Valid-Flag



Key Management: MIKEY-Modes

MIKEY defines 3 options for the authentication and negotiation of session keys

(All as 2 way handshakes):

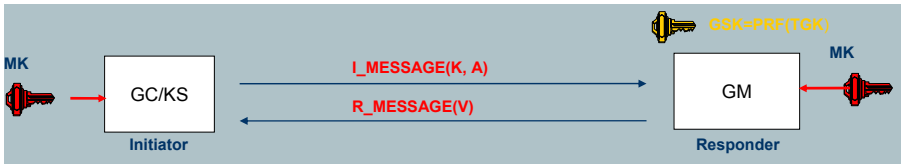
- Symmetric key distribution (pre-shared keys, MAC for integrity protection)
- Asymmetric key distribution (based on asymmetric encryption)
- Diffie Hellman key agreement protected by digital signatures
  - Creates a DH-key, which is used as the TGK
  - cannot be used to create group keys; only single peer-to-peer keys

Two further versions exist, which are not part of RFC3830 itself

- Diffie Hellman key agreement protected by symmetric pre-shared keys
- Asymmetric key distribution (based on asymmetric encryption) with in-band certificate provision

- The default and mandatory key transport encryption is **AES in counter mode** [RFC3711]
- The default and mandatory keyed hash algorithm is **HMAC-SHA-1**

Key Management: MIKEY – Symmetric key distribution



Initialization:

Rand, TGK:= Random()

encr-key, auth-key := PRF(MK,…|| Rand)

Protocol execution:

K := [ID_A] || [ID_B] || T || Rand || $E_{encr-key}$(TGKs [|| KEK]) || {SP}

A := HMAC-SHA1(auth-key, K)

Retrieve **TGK** from K

auth-key := PRF(MK, …|| Rand)

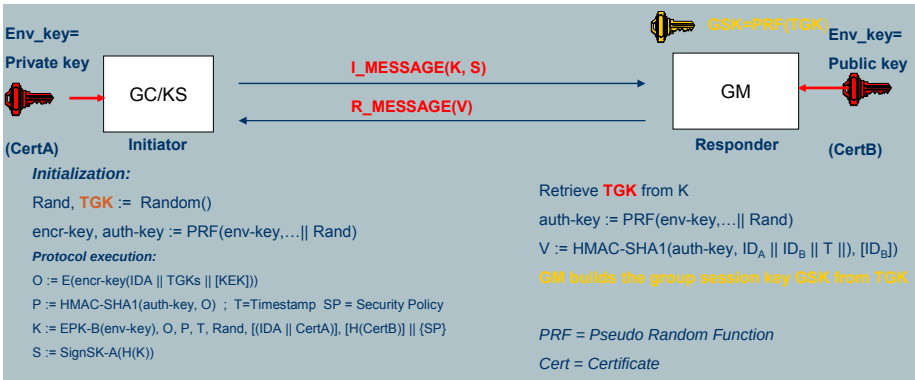
V := HMAC-SHA1(auth-key, ID_A || ID_B || T ||), [ID_B])

GM builds the group session key GSK from TGK

PRF = Pseudo Random Function

- **Pre-shared secret based distribution**
- May proceed as 2-way handshake (optional second message)
- Only the initiator influences the key generation
- No PKI support necessary

Key Management: MIKEY – Asymmetric key distribution



- May proceed as 2-way handshake (optional second message)
- Only the initiator influences the key generation
- PKI support necessary

Key Management: Reasons for Rekeying / Key lifetime

- The group key may need to be changed on demand
  - if it is determined that the key has been compromised
- Maximal lifetime of a key depends on many factors, e.g. used algorithms
- Example key lifetime of 2^{32} PTP-packets:
  - 1000 packets/sec (worse case: short sync with 1ms interval)
  - results in max. 50 days of communication
- A rekeying has to be done before the end of lifetime !
- A rekeying has to be done after N PTP-Sync-Messages
- N should be << 2^{32} packets

Key Management: Rekeying Protocol (2)

- The rekey protocol periodically updates or changes the Group session key (GSK)
- The group members can request re-synch at the GC/KS

  (if their keys expired and an updated key has not been received)
- For a synchronous key change, the rekeying will be done in 2 steps:

  1.) distribute new Goup Session Key (GSK) (and get acknowledged)

  2.) include information on switch-over time
  using the **sequence number of PTP sync_msg**

  E.g.: the new session key is valid starting with PTP_sequence_number 751
- MIKEY: GC/KS: Send I_MESSAGE(encypted(GSK), PTP_sequence_number=751)
  - GMs: Send R_MESSAGE(Valid-Flag)
- In MIKEY rekeying protocol is the registration protocol. Intiator is the GC/KS
- **to avoid implosion problems in large scale installations the rekey message can be sent in multicast (push). This requires that all group members use the same master Key (MK). The Acknowledge mustn't be requested (GC overload). If a GM didn't get the new key, it can request it separately at the GC/KS (pull)**

Key Management: Rekeying Protocol (3)

- Rekeying Message:
  The Group Controller distributes the new GSK to all GMs
- rekeying starts in configured time intervals
- Synchronous activation of the new key after complete distribution
- A sequence number of the PTP-sync_message will be distributed with the rekeying_message
- For efficiency the "MIKEY symmetric key distribution scheme" shall be used for rekeying

Conclusion

For securing the PTP-Protocol the two presented approaches have to be implemented:

- A Group Shared Key (GSK) for message integrity protection
- A Key management is required for larger installations of PTP-netwoks

Next Steps:

Specify the details for transport of MIKEY over PTP:

- Specify the Security Policy Parameters to be transferred in the I_MESSAGE:
  - Hash algorithm to use , Key length
  - Activation timepoint for the new key (use Ptp_sequence_number)
  - Key lifetime (use Ptp_sequence_number)
- Enable Fault tolerance for GC/KS: Locate GC/KS in PTP-Master
  - Dynamically changing master role has to be considered for Registration with pairwise shared secrets

End

Thank You

Presented by: Stephan Schüler

Siemens Communications

Schueler.stephan@siemens.com

Backup

Abstract

The PTP-Protocol has no security mechanism to protect the transmission of PTP-messages in the current revision of the standard IEEE1588-2002. But there is a strong demand to extend the standard by security mechanism since the most customers already have their security policies which have to be fulfilled if they want to introduce new applications which include the PTP-Protocol. The security policies result from known threats as there are passive and active attacks like eavesdropping, man-in-the middle-attacks or replay-attacks.

The presentation gives an overview for countermeasures to these threats which should be defined in the next revision of the IEEE1588 standard, especially mechanisms for authentication, authorization and integrity protection. The currently active P1588 working group is already taking these requirements into account.

After the overview, a concrete proposal for a security extension will be presented. It covers two main countermeasures: The first one is the integrity protection and authentication of the PTP-messages between the PTP-Ports and the second one is a mechanism for a role based access control which is required for the secure administration of PTP-Ports.

The presented solution describes the required elements for a message extension, concrete algorithms which are suitable for the integrity protection and authentication and the authorization mechanism which is based on shared secrets.

FINAL PARTICIPANTS

Werner Abt
Dipl. Ing.
IXXAT Automation Gmbh
Leibnizstrasse 15
Weingarten 88250, Germany
+49-751-561-46-63
+49-751-561-46-29
abt@ixxat.de

Galina Antonova
Design Engineer
GE Consumer & Industrial
8525 Baxter Place, Suite 100
Burnaby, V5A 4V7, Canada
1 604-421-8629
galina.antonova@ge.com

Douglas Arnold
Chief Scientist, Symmetricom, Inc.
3750 Westwind Blvd.
Santa Rosa, CA 95403 USA
1-707 636 1929
darnold@syymmetricom.com

Sivaram Balasubramanian
Senior Research Engineer
Rockwell Automation
1 Allen Bradley Drive
Mayfield Heights, OH 44124 USA
sbalasubramanian@ra.rock

Nick Barendt
Software Engineer
VXI Technology, Inc.
5425 Warmer Road, Suite 13
Valley View, OH 44125 USA
216-447-8951
nbarendt@vxitech.com

Dominic Béchaz
Dipl. IT Ing. FH
ZHW Technikumstrasse 9
Winterthur 8401 CH
+41522677148
bdo@zhwin.ch

Stephan Bedrosian
DMTS, Agere Systems
300 Brickstone Square
Andover, MA 01810
USA
978-691-3017
psbedrosian@agere.com

Michael Branicky
Professor
Case Western Reserve University
Electrical Engineering and Computer Science
10900 Euclid Ave.,
Glennan 517B
Cleveland 44106 United States
216-368-6039
mb@case.edu

Chirstoph Brunner
ABB Schweiz

Stewart Bryant
Technical Leader
Cisco Systems
250 Longwater Avenue
Creep Park
Reading RG2 6GB, UK
+44 208 824 8828
stbryant@cisco.com

Michael Buckel
ABB Schweiz
Austrasse, Turgi
5300 Schweiz

Dr. Stefan Burkhardt
Development Engineer
Rohde & Schwarz Gmbh & Co KG
Muehldorfstr.15
Muenchen 81671 Germany
+49 (89) 4129-14317
+49 (89) 4129-64317

Martin Burnicki
Meinberg Funkuhren
Auf der Landwehr 22
Bad Pymont
D-31812, Germany
Martin.burnicki@meinberg.de

Christopher Calley
Technologist, Symmetricom
3750 Westwind Blvd.
Santa Rosa, CA 95403 USA
707-636-1821
ccalley@symmetricom.com

Ron Cohen
CTO, Resolute Networks
Ligad Center
15 Hashdera Hamerkazit
Modi'n 71700 Israel
+972-52-8995764
ronc@resolutenetworks.com

Kendall Correll
Intern, VXI Technology, Inc.
5425 Warmer Road, Suite 13
Cleveland, OH 44125 USA
216-447-8950
kendallc@vxitech.com

Augusto Ciuffoletti
Prof. INFN
Via Milano 118-38100 Trento
1-38100 Italy
0461-923772
0461-923772
augusto@di.unipi.it

Gregorie Diehl
Fachinformatiker
Meinberg Funkuhren
Auf der landwehr 22
Bad Pyrmont
D-31812, Germany
052 81 93 09 29
gregorie.diehl@meinberg.de

Peter Duchemin
Smart Network Devices
Karl-Heinz-Beckurts-Strasse 13
Juelich D-52428
pduchemin@smartnd.com

John Eidson
Department Scientist
Agilent Technologies
3500 Deer Creek
Rd.MS-24MA
Palo Alto, CA 94304 USA
650-485-4263
650-485-3894
john_eidson@agilent.com

Jean-Loup Ferrant
Standardisation Manager
Alcatel, Villarceaux
Nozay 91625, France
33 1 6449 2307
33 1 6449 2956
jean-loup.ferrant@alcatel.com

Ulrich Fiedler
Senior Researcher
ETH Zurich
Gloriastr.35
Zurich 8092, CH
+41 44 632 70 09
+41 44 632 10 35
fiedler@tik.ee.ethz.ch

John Fischer
VP Engineering
Spectracom Corp
95 Methodist Hill Drive, Suite 500
Rochester, NY 14623 USA
001-585-321-5800
001-585-321-5219
jfischer@sprectracomcorp.com

John Fleck
Senior Software Engineer
TTC
15 Terry Drive
Newtown, PA 18940 USA
352-2020 x182
267-352-2021
jfleck@ttcdas.com

Georg Gaderer
Reasercher
Austrian Academy of Sciences
Viktor Kaplan Strabe 2
Wiener Neustadt 2700
Austria
+43-676-843-104-650
georg.gaderer@tuwien.ac.at

Geoffrey Garner
Consultant, Samsung
196 Ambassador Drive
Red Bank, NJ 07701 – USA
+1 737 758 0335
gmgarner@comcast.net

Michael Gerstenberger
Senior Engineer
KUKA Robotics
22500 Key Drive
Clinton Township 48036, USA
248.844.0561
michaelgerstenberger@kukarobotics.com

Heiko Gerstung
Sftware-Entwickler
Meinberg Funkuhren
Auf der Landwehr 22
Bad Pyrmont
D-31812 Germany
052 81 9309-25
heiko.gerstung@meinberg.de

Clemens Gmeiner
KUKA Robotics

Franz-Josef Götz
Dipl-Ing, Siemens AG
Gieiwitzerstrasse 555 Nümberg
90475 Germany
+499118953455
+499118953715
franz-josef.goetz@siemens.com

John Guilford
Member of the Technical Staff
Agilent Technologies
3500 Deer Creek
Rd.MS-24MA
Palo Alto, CA 94304 USA
650-485-4263
650-485-3894
john_guilford@agilent.com

Bruce Hamilton
MTS, Agilent Technologies
P.O.Box 10350
Palo Alto, CA 94025 USA
+1-650-485-2818
bruce_hamilton@agilent.com

Ken Harris
Principle Engineer
Rockwell Automation
1 Allen-Bradley Dr.
Mayfield Heights, Ohio 44060 – USA
440-646-3621
440-646-3258
krharris@ra.rockwell.com

Christoph Heitz
Prof. Dr.
ZHW
Jägerstrasse 2
Winterthur 8401 Switzerland
++41-52-2677816
christoph.heitz@zhwin.ch

Oyvind Holmeide
Managing Director
Westermo OnTime AS
Gldsvei 20 Osio 0489 Norway
+47 22090304
+47 22090310
cyvind@ontimenet.com

Gerhard Hübner
Geschäftsleitung, Symmetricom GmbH
Fichtenstarsse 25
Hofolding 85649 Germany
08104 662415
ghuebner@symmetricom.com

Harrell Huckeba
Technologist, Symmetricom, Inc.
3750 Westwind Blvd.
Santa Rosa, CA 95403 USA
707 636 1937
hhuckeba@symmetricom.com

Keen Hun Leong
Senior R&D Engineer
Agilent Technologies
Phase III, Bayan, Lepas Free
Industrial Zone, Bayan Lepas 11900
Malaysia
+60 (4) 680-7922
+60 (4) 645-6604
Keen-hun_leong@agilent.com

Drew Jenkins
Maxim Integrated Products

Steven Jennings
R&D Engineer
Bosch Rexroth
Partensteinerstr.23
Laoh am Main 97816, Germany
+49 9352 18 1772
steven.jennins@boschrexroth.de

Blattner Joerg
Development Engineer Realtime Coomunication Systems
Zurich University of Applied Sciences
Technikumstrasse 9
Winterthur 8401 Switzerland
+41 52 276 76 87
bjo@zhwin.ch

Frank Kardel
Meinberg Funkuhren
Auf der Landwehr 22
Bad Payrmont
D-31812, Germany
Frank.kardel@meinberg.de

Joel Keller
Software Engineer
National Instruments
11500 North Mopac
Austin 78759 USA
512-683-8840
joel.keller@ni.com

Nikolaus Kerö
CEO Oregano Systems
Phorusgasse 8
Vienna 1040, Austria
+43 676 843 104-300
nikolaus.keroe@oregano.at

Lean Kim Ong
Hardware Design Engineer
Agilent Technologies
Bayan Lepas Free
Industriel Zone, Bayan Lepas, Penang 1190
Malaysia
604-680-7224
lean-kim_ong@agilent.com

Michel Korreng
R&D Engineering Manager
Symmericom
3750 Wetwind Blvd.
Santa Rosa, CA 95403 USA
1-707-636-1801
mkorreng@symmetricom.com

Ulrich Kunkel
R&D Engineer
Hottinger Baldwin Messtechnik
Im tiefen See 45
Darmstadt 64293, Germany
00496151803546
00496151803524
ulrich.kunkel@hbm.com

Patrick Lagrange
System Engineer, Nortel
Rue paul cezanne
Guyancourt 78942, France
+33 1 39 44 56 45
lagrange@nortel.com

Gordon Lau
Hardware Designer
ruggedCom Inc.
64 Jardin Drive
Unit 3G Concord L4K3P3 Canada
905-760-7709
905-760-9909
gordonlau@ruggedcom.com

Kang Lee
Group Leader, NIST
100 Bureau Drive, MS#8220
Gaithersburg, MD 20899-8220, USA
301-975-6604
301-990-3851
kang.lee@nist.gov

Richard Lee
iPosi, Inc.
Greenwood Village
CO 80111
South Chester Way 6095, USA
303 267 0159
303 267 0181

Ya-Shian Li
Computer Scientist
National Institute of Standards and Technology
100 Bureau Drive MS 8120
Gaithersburg, MD 20899, USA
1-301-975-5319
ya-shian.li@nist.gov

Ray Lock
Technical Director
Westermo Data Communications
Talisman Business Centre
Park Gate Southampton SO31 7GA UK
0044 7787522703
raylock@westermo.co.uk

Udo Maltzahn
Entwicklung
Meinberg Funkuhren
Auf der Landwehr 22
Bad Pymont
D-31812 Germany
052 81 93 09-19
udo.maltzahn@meinberg.de

Alex McCarthy
PXI & Instrument Control Group Manager
National Instruments
11500 North Mopac
Austin, 78759, USA
512-683-5310
alex.mccarthy@ni.com

Hung Mach
Design Engineer
Boeing
P.O. Box 3707
MS 14-ME
Seattle, WA 98124 USA
(206) 655-9926/206-655-7724
hung.k.mach@boing.com

John Mackay
Engineer
Progeny Systems
9500 Innovation Drive
Manassas, VA 20110 – USA
703-368-6107 x144
john.mackay@progeny.net

Christophe Masson
EMEA Telecom product manager
Zarlink 10 av. Du Quebec
APIS – Bat F3
Courtaboeuf 91944 France
+33 160 92 41 94
christophe.masson@zarlink.com

Sven Meier
Development Engineer
ZHW Eichenweg
8 Wettsvil a. A. 8907 Switzerland
+41796716211
msv@zhwin.ch

Anatoly Moldovansky
Principal Engineer
Rockwell Atuomation
1 Allen-Bradley Dr.
Mayfield Heights, Ohio 44124 – USA
440-646-5369
440-646-3076
amoldovansky@ra.rockwell.com

Laurent Montini
Consulting System Engineer
Cisco Systems
11, rue Camille Desmoulins
Issy-les-Moulineaux
92782 France
+33 (0) 5804 6453

Dirk Mohl
Dipl.-Ing. (FH)
Hirschmann Automation and Control
Stuttgarterstr.45-51
Neckartenzlingen 72654, Germany
++49 7127 141622
dirk.mohl@hirschmann.de

Prof. Thomas Müller
ZHW Institute of Embedded System
Technikumstrasse 9
Winterthur 8400 Schweiz
052 267 75 09
mth@zhwin.ch

Gabriel Narus
Hardware Engineer
National Instruments
11500 North Mopac
Austin 78759 USA
512-683-8908
Gabriel.narus@ni.com

Mike Neering
Director, New Business technology
Agilent Technologies, Inc.
1400 Fountaingrove Pkwy.
M/S 2USA, Santa Rosa 95403
707-577-2756/707-577-4562
mike_neering@agilent.com

Karen O'Donoghue
Lead Engineer US Navy
Code B35
17320 Dahlgren Road
Dahlgren 22448 USA
+1 540 653 1567
+1 540 653 8673
karen.odonoghue@navy.mil

Rob Rennard
Marketing Manager
Agilent Technologies, Inc.
1400 Fountaingrove Pkwy.
M/S 2USA, Santa Rosa 95403
707-577-3140/707-577-4562
bob_rennard@agilent.com

Dave Roe
Semtech Ltd.
Units 2 & 3 Park Court, Premier Way
Abbey Park Industrial Estate
Romsey, Hampshire SO52 9DN – UK
01794 527600
01794 527601
droe@semtech.com

Silvana Rodrigues
System Architect
Zarlink Semiconductor
400 March Road
Ottawa K2K 3H4 Canada
+16132707590
+16135921010
silvana.rodrigues@zarlink.com

Paul Rushton
Semtech, Ltd.
Units 2 & 3 Park Court, Premier Way
Abbey Park Industrial Estate
Romsey, Hampshire SO52 9DN – UK
01794 527600
01794 527601
prushton@smtech.com

Markus Seehofer
Dipl.-Ing. (FH)
Hirschmann
Stuttgarterstr.45-51
Neckartenzlingen 72654, Germany
++497 127 141834
markus.seehofer@hirschmann.de

Thomas Schenkel
Project Leader
KOCH IT AG
Rudolf Diesel-Strasse 5
Winterthur 8404 CH
052 235 08 35
052 235 08 36
thomas.schenkel@koch-it.ch

Wollfgang Schmid
Dipl.-Ing. (FH)
Hirschmann Automation and Control
Stuttgarterstr.45-51
Neckartenzlingen 72654, Germany
++49 7127 141910
++49 7127 141600

Steffen Schwips
R&D Director
Jetter AG
Graeterstr.2
Ludwigsburg, 71642, Germany
+49-7141-2550-0
sschwips@etter.de

Joachim Stolberg
Dipl. Ing. / Project Manager
IXXAT Automation Gmbh
Leibnizstrasse 15
Weingarten 88250, Germany
+49-751-561-46-23
+49-751-561-46-29
abt@ixxat.de

Dominik Schneuwly
Senior Engineer, Oscilloquartz
Brevards 16 Neuchaetel CH-2000
Switzerland
+41 32 722 55 67
+41 32 722 55 56
schneuwly@oscilloquartz.com

Nicolas Voirol
R&D Manager, Oscilloquartz
Brevards 16 Neuchaetel CH-2000
Switzerland
+41 32 722 55 75
+41 32 722 55 56
voirol@oscilloquartz.com

Nirmala Shenoy
Associate Professor, Information Technology
Rochester Institute of Technology
102 Lomb Memorial Drive
Rochester, NY 14623 USA
001 585 475 4887
001 585 475 2181
ns@it.rit.edu

Stephan Schüler
Dipl.-Ing.
Siemens AG Brauckstr.14
Witten 58449 Germany
+49 (0) 2302 667 2063
+49 (0) 2302 667 3228
schueler.stephan@siemens.com

Paul Skoog
Product Marketing Manager
Symmetricom
3750 Westwind Blvd.
Santa Rosa, CA 95403 USA
707-636-1955
pskoog@symmetricom.com

Samuel Stein
President
Timing Solutions Corporation
4775 Walnut St
Suite 1 B
Boulder, Colorado 80301 USA
303-939-8481
srstein@timing.com

Christoph Thurnheer
Development Engineer
ZHW Technikumstrasse
Winterthur 8524 Switzerland
052 267 76 78
thc@zhwin.ch

Phil Tolcher
Semtech, Ltd.
Units 2 & 3 Park Court, Premier Way
Abbey Park Industrial Estate
Romsey, Hampshire SO52 9DN – UK
01794 527600
01794 527601
ptolcher@semtech.com

Dave Tonks
Semtech, Ltd.
Units 2 & 3 Park Court, Premier Way
Abbey Park Industrial Estate
Romsey, Hampshire SO52 9DN – UK
01794 527600
01794 527601
dtonks@semtech.com

Jeanette Tsang
Student, University of British Columbia
49 East 58th Avenue
Vancouver, V5X 1V7 Canada
1-604-322-2820
jsmtsang@gmail.com

Dietrich Vook
Signal Processing Group Manager
Agilent Laboratories/Agilent Technologies
26M-3 3500 Deer Creek Rd
Palo Alto, CA 94304 USA
(650) 485-5244
(650) 4850-8950
dieter_vook@agilent.com

Alijosa Vrancic
Senior Software Engineer
National Instruments
11500 North Mopac Expressway, Bldg. C
Austin 78759 USA
512-683-8000
alijosa.vrancic@ni.com

Karl Weber
Senior Engineer
Beckhoff
Eiserstr.5
Verl
33415 Germany
+49 151 1633 5381
k.weber@beckhoff.com

Prof. Hans Weibel
Prof. ZHW
Technikumstrasse 9
Winterthur 8401 Switzerland
+41 52 267 75 52
hans.weibel@zhwin.ch

Matthias Wenk
Siemens AG
A&D MC RD 15
Frauenauracher Str. 80
Erlangen 91056
+49 9131-982258
+49 9131-981130
mathias.wenk@siemens.com

Ludwig Winkel
Fieldbus Standardization Manager
Siemens A&D PT2 FC
Siemensallee
73 Karisruhe 76187 Germany
+4972159560988
+4972159893609

Jochen Wolle
Head of R&D Software Rohde & Schwarz
Muehldorfstr 15
Munich 81671 Germany
++49 89 4129 13044
jochen.wolle@rsd.rohde-schwarz.com