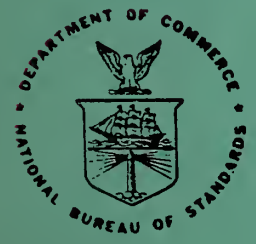
NBSIR 85-3236

An NBS Host to Front End Protocol

C. Michael Chernick

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Systems and Network Architecture Division
Gaithersburg, MD 20899

August 1985



U.S. DEPARTMENT OF COMMERCE
BUREAU OF STANDARDS

NBSIR 85-3236

AN NBS HOST TO FRONT END PROTOCOL

C. Michael Chernick

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Systems and Network Architecture Division
Gaithersburg, MD 20899

August 1985

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

An NBS Host to Front End Protocol

C. Michael Chernick

9 August 1985

Table of Contents

List of Figures

Part I - Introduction to the Description

1 Introduction

The National Bureau of Standards (NBS) Computer Networking Program has specified a host-to-front end communications protocol intended to be used in researching networking protocol performance. This protocol was developed not as a proposed standard, but rather to support protocol performance testing. This report is being published to document the protocol performance group's research and development into host-to-front end communications protocols so that it might prove useful to others wishing to implement this or similar protocols.

2 Organization of the Description

This description of the Host-to-Front End Protocol (HFEP) is divided into three parts. This, Part I, describes the background, purpose, and design goals of the HFEP. Part II is the HFEP Service Specification, which describes the services that the protocol offers to a user. The last part, Part III, is the HFEP Specification which describes the protocol mechanisms themselves using a formal description technique [Tenn81], time sequence diagrams, state tables and header formats for the protocol data units (PDU's) employed.

3 Background, Purpose, and Design Goals

Many modern computer communications protocols are complex to design and build and require large amounts of host computer resources (e.g., memory, cpu time, system services). Complexity in designing, building, and tuning these protocols can often lead to delay in problem solving using networking, especially in view of changing technology and communications policy (e.g., tariffing). The resource loads imposed by communications software on a host can be quite large, possibly significantly reducing the potential computer networking gains.

One method that can be used to reduce the impact of these drawbacks is the use of so called "Front End" computing systems (also called "Outboard Processing Elements" (OPE) [Padi84]). Such systems are usually connected to host computers through very reliable high bandwidth computer communications channels. They off-load many of the tasks involved with maintaining connections over numerous, possibly highly unreliable, communications networks. Modern technology, such as single board communications oriented computers, allows the development of these OPE's at a reasonable cost. Complex communications protocols can be programmed into a relatively small number of ROM chips. Already

commercial implementations of the ISO transport class 4 [IS8073] are available on small computers at modest costs. More implementations supporting a wider variety of underlying network technology and higher layer protocols can be expected in the future.

The NBS computer Networking Program is concerned with both the performance of communications protocols themselves and with their effect on host computer systems. We are also concerned with testing methodology for protocol performance measurement. For these reasons we see merit in using OPE's both in direct support of our research and as a general tool for the networking research community.

However, before an OPE can be connected to a host computer, a host-to-front end protocol (HFEP) must be developed. This protocol should be flexible enough to allow the off-loading of a wide variety of communications protocols and yet still be reasonably simple to avoid overloading the host computer with tasks possibly just as complex as those off-loaded. (M. Padilipsky comments further on this problem in a discussion of the concept of "advantageous to implement" [Padi84].) Since the path from the host to the OPE is normally extremely reliable (almost on a par with intra computer path reliability), the HFEP need not be overly concerned with host to OPE communications errors. This greatly simplifies HFEP design and implementation.

To further reduce design effort and enhance capability it was decided early in the HFEP development (during the Fall of 1984) to base the design as much as possible on available standards and products. Thus the current design is based on X.25 technology to provide the reliable connection oriented service needed for the host to OPE communications path. A factor in this choice was the existence of a commercially available X.25 device for our host computer (a VAX 780). However, while our design is based primarily on X.25, other reliable, multiplexed and individually flow controlled network connection oriented technologies could be used in place of X.25.

Part II - HFEP Service Specification

1.0 Introduction

Some of the terminology and concepts used in this part of the description are the same or similar to those used in the ISO Open System Interconnection (OSI) Basic Reference Model (BRM) [IS7498]. It is suggested that the reader become familiar with the material in the BRM.

The Host Front End Protocol (HFEP) provides a reliable connection oriented process-to-process communications path between a host computer and a front end computer (also called an Outboard Processing Element (OPE) [Padil84]). The HFEP is symmetric i.e., the same service is provided to the OPE as to the host. While it provides service that can be used by an upper layer protocol, HFEP is independent of any particular upper layer protocol such as ISO Transport. These services are provided by mapping to a lower layer protocol (in this case X.25). In addition, HFEP provides other functionality such as process rendezvous (initial connection service). Possible future extensions to HFEP service could be HFEP connection testing (via loopback) and HFEP traffic monitoring.

Since the communications path between the host and OPE is relatively short and link level error detection and correction is provided by X.25, HFEP does not provide for error correction and recovery although error detection and reporting are included. (It may be decided that underlying X.25 services are reliable enough so that even this is not necessary.) It is anticipated that either the host to OPE link will be essentially error free (after X.25 link level error recovery) or the link is so error prone as to be useless. In either case, error recovery techniques above and beyond X.25 hardly seem beneficial and would be detrimental to the design goals of offloading the host and providing high host to OPE throughput.

2.0 Types Of Service Primitives

Service provided is described in a manner similar to that used for Transport [ICST83]. There are four types of primitives, namely:

a) request (issued by HFEP user to request HFEP service),

b) indication (issued by HFEP to indicate an action),

c) response (issued by HFEP user to respond to an indication), and

d) confirmation (issued by HFEP to confirm some request).

3.0 HFEP Services

3.1 HOPEN Service

The HOPEN service is used to establish an HFEP connection and to transfer a limited amount of user supplied data in each direction. HOPEN is a confirmed service.

3.1.1 HOPEN.request(uconnid,lhsap-id,fhsap-id,udata)

The HOPEN.request is issued by a user of HFEP to attempt to establish an HFEP connection from a local HFEP Service Access Point (HSAP) to a foreign HSAP. (Note: The concept of Service Access Points is discussed in the OSI Basic Reference Model [IS7498]). The user may include a limited amount of data with the HOPEN request.

uconnid - a non-zero identifier assigned by the local user of the potential HFEP connection to distinguish this connection request from any others. (This identifier has local significance only.)

lhsap-id - the local HSAP identifier (an address) which has end to end significance between the host and the front end (OPE). This is the address from which a connection is being requested.

fhsap-id - the foreign HSAP identifier (an address) which has end to end significance between the host and the front end (OPE). This is the address to which a connection is being requested.

udata - a limited sequence (up to 32 octets) of user supplied data to be delivered by the corresponding HOPEN indication to the foreign HSAP identified by fhsap-id.

3.1.2 HOPEN.indication(uconnid,hconnid,lhsap-id,fhsap-id,rudata)

The HOPEN.indication is issued by an HFEP entity to an HFEP user to indicate the arrival of a connection request. The HFEP user must have previously issued an HLISTEN.request (See HLISTEN Service below.) to the local HFEP entity to indicate the user's willingness to accept HOPEN.indications. Two possibilities exist. Either the previous HLISTEN explicitly indicated a local HSAP identifier whose value was lhsap-id or the previous HLISTEN specified a local HSAP identifier of 0 to

indicate it's willingness to accept HOPEN.requests for any local HSAP identifier. The HFEP entity will only issue an HOPEN.indication to a user that has HLISTENed with a local HSAP identifier of 0 in the case where no other user has explicitly issued an HLISTEN indicating the same local HSAP identifier. The HOPEN.indication may include a limited amount of user data.

uconnid - an identifier originally assigned by the local user and included in a HLISTEN.request for the potential HFEP connection. This identifier distinguishes this connection indication from any others. (This identifier has local significance only.)

hconnid - an identifier assigned by the local HFEP entity to distinguish this potential HFEP connection indication from any others. (This identifier has local significance only.)

lhsap-id - the local HSAP identifier (an address) which has end to end significance between the host and the front end (OPE). This is the address for which a remote connection was requested.

fhsap-id - the foreign HSAP identifier (an address) which has end to end significance between the host and the front end (OPE). This is the address from which a connection was requested.

rudata - a limited sequence (up to 32 octets) of user supplied data included with the corresponding HOPEN.request.

3.1.3 HOPEN.response(hconnid,udata)

The HOPEN.response is issued by a user of HFEP to indicate his willingness to accept a connection. A previous HOPEN.indication must have arrived indicating hconnid as the HFEP assigned connection identifier. an HFEP connection from a local HSAP (HFEP Service Access Point) to a foreign HSAP. The user may include a limited amount of data with the HOPEN.response.

hconnid - an identifier assigned by the local HFEP entity to the potential HFEP connection to distinguish it from any others. (This identifier has local significance only.)

udata - a limited sequence (up to 32 octets) of user supplied data to be delivered by the corresponding HOPEN.confirm to the foreign HSAP.

3.1.4 HOPEN.confirm(uconnid,hconnid,rudata)

The HOPEN.confirm is issued by an HFEP entity to an HFEP user to indicate the arrival of a connection confirm. The HFEP user must have previously issued an HOPEN.request to the local HFEP entity to indicate the user's desire to initiate a HFEP connection. The HOPEN.confirm may include a limited amount of remote user data.

uconnid - an identifier originally assigned by the local user and included in a HOPEN.request for the potential HFEP connection. This identifier distinguishes this connection indication from any others. (This identifier has local significance only.)

hconnid - an identifier assigned by the local HFEP entity to distinguish this potential HFEP connection indication from any others. (This identifier has local significance only.)

rudata - a limited sequence (up to 32 octets) of remote user supplied data included with the corresponding HOPEN.response.

3.2 HLISTEN Service

The HLISTEN service is associated with the HOPEN service and is used to inform the local HFEP entity that the issuer wishes to be notified (via an HOPEN.indication) of arriving HOPEN connection requests. (See Appendix A for a discussion of the N + 1 Layer rendezvous problem.) HLISTEN is of local significance only since no HFEP Protocol Data Units (PDUs) can be sent directly as a result of an HLISTEN.request. HLISTEN.request is the only service primitive associated with the HLISTEN service.

If for some reason (perhaps lack of local resources) the HLISTEN.request cannot be honored by the local HFEP entity, an HCLOSE.indication with a user connection identifier of uconnid will arrive subsequently.

3.2.1 HLISTEN.request(uconnid,lhsap-id)

The HLISTEN.request is issued by a user of HFEP to indicate his willingness to accept an indication for a connection. The local HFEP entity places the lhsap-id in its internal tables. When a subsequent connection request arrives for this lhsap-id, the HFEP entity can determine to which local user the indication should be delivered. As a special case, lhsap-id can be 0 to

inform the local HFEP entity that this local user will accept delivery for any lhsap-id.

uconnid - a non-zero identifier assigned by the local user of the potential HFEP connection to distinguish this HLISTEN.request from any others. This identifier will be returned to the HFEP user with any subsequent HOPEN.indications that arrive. (This identifier has local significance only.)

lhsap-id - the local HSAP identifier (an address) which has end to end significance between the host and the front end (OPE). A future HOPEN.request for lhsap-id can be delivered to this user. If the value of lhsap-id is 0 this user will accept an indication for any local HSAP.

3.3 HDATA Service

The HDATA Service is the primary method of transmitting and receiving HFEP user data across the host-front end boundary. HDATA is a non-confirmed service consisting of two primitives, HDATA.request and HDATA.indication. An HFEP connection must have been previously established before user data can be sent or received.

The data stream is composed of a sequence of HFEP service data units (HSDU's). Each HSDU is itself a sequence of user supplied octets. The end of an HSDU is marked by an END OF HSDU (EOHSDU) flag. User data may (but need not) be buffered at the transmitting (or receiving) HFEP entity until an EOHSDU is indicated. The HFEP HDATA service guarantees that data will be delivered in order. That is, the order that a user presents data to HFEP via an HDATA.request primitive at one end of an HFEP connection is the same order that data are delivered at the other end of the connection by an HFEP.indication. Both the sequence of octets within an HSDU and the sequence of HSDU's in an HFEP connection is preserved.

Often an entire HSDU is not sent with one HDATA.request but rather is sent with several HDATA.requests. For example, an HSDU of 300 octets could be sent as one HDATA.request marked with the EOHSDU flag, or it could be sent as 2 HDATA.requests of 100 octets each followed by an HDATA.request of 100 octets marked with the EOHSDU flag. Data delivered to the user at the receiving end may be fragmented differently than that sent to the transmitting end by the transmitting user. Thus an HSDU of 300 octets originally sent as a single HDATA.request of 300 octets marked with the EOHSDU flag could be delivered with one HDATA.indication of 300 octets marked with the EOHSDU flag, or it could be delivered as 2 HDATA.indications of 100 octets each followed by an HDATA.indication of 100 octets marked with the EOHSDU flag.

However, fragmentation never occurs across HSDU boundaries. Thus, data that must be kept logically intact across the HFEP interface must be sent within a single HSDU.

3.3.1 HDATA.request(hconnid,udata,eohsduf)

The HDATA.request primitive is issued by an HFEP user to transmit data to the other side of a previously opened (via HOPEN primitives) HFEP connection. User data may be buffered by the transmitting side if the eohsduf (end of HSDU flag) is false.

hconnid - an identifier returned by one of the HOPEN primitives to uniquely indicate which connection is to be used for the data transmission.

udata - the user data to be transmitted. Udata is a (non-null) sequence of octets the contents of which are completely ignored by the HFEP entities (i.e., data transparency is provided).

eohsduf - a flag which if set indicates that this HDATA.request contains the last (and perhaps only) user data that is to be delivered to the remote HFEP entity as part of the current HSDU. If eohsduf is false, this HDATA.request contains either the initial or a middle portion of an HSDU. If eohsduf is false, data may not actually be transmitted or delivered to the remote HFEP user until a subsequent HDATA.request is issued with eohsduf set true.

3.3.2 HDATA.indication(hconnid,rudata,eohsduf)

The HDATA.indication primitive is used to indicate to an HFEP user that data has arrived from the remote side. The connection must have previously opened using the HOPEN service. Data delivered to the user by the local HFEP entity may not necessarily be of the same sequence length as that delivered to the remote HFEP entity due to fragmentation by the HFEP service. However, if the eohsduf (end of HSDU flag) is set, then this indication is the last (or perhaps only) HDATA.indication for an HSDU sent by the remote user.

hconnid - an identifier returned by one of the HOPEN primitives to uniquely indicate which connection is to be used for the data indication.

rudata - the user data being delivered. Rudata is a (non-null) sequence of octets the contents of which are completely ignored by the HFEP entities (i.e., data transparency is provided).

eohsduf - a flag which if set indicates that this HDATA.indication contains the last (and perhaps only) user data that is to be delivered to as part of the current HSDU. If eohsduf is false, this HDATA.indication contains either the initial or a middle portion of an HSDU. If eohsduf is false, additional remote user data is yet to be delivered as part of the current HSDU.

3.4 HCLOSE Service

The HCLOSE service is used to terminate the use of an HFEP connection. It is also used to indicate that one side of an "opening" connection (a connection in the process of being established) refuses to open the connection. It is a nonconfirmed service. Any user data that has been previously sent using the HDATA.request primitive may be lost. After issuing an HCLOSE.request or receiving an HCLOSE.indication for a given connection, a user may not issue further HDATA.requests nor expect to receive any further HDATA.indications. A user specified disconnection reason (an unsigned binary number) and a limited amount of user data may be sent with the HCLOSE service.

3.4.1 HCLOSE.request(uconnid,hconnid,ureason,udata)

The HCLOSE.request primitive requests the removal of an HFEP connection, the refusal of one side to open a connection (in the case where a HOPEN.indication has been received) or the cancellation of an HLISTEN.request for the connection identifier uconnid.

uconnid - an identifier previously assigned (in an HOPEN.request or an HLISTEN.request) by the local user of the HFEP to distinguish this connection (or potential connection) from any others.

hconnid - an identifier assigned by the local HFEP entity to distinguish this connection (or potential connection) from any others.

ureason - an unsigned binary number specifying the user's reason for closing or refusing the connection. Ureason is limited to 16 bits. This number will be delivered to the other side of the (perhaps potential) connection with the HCLOSE.indication primitive.

udata - a limited sequence of (up to 32) octets of user supplied data to be delivered to the other side of the (perhaps potential) connection with the HCLOSE.indication primitive.

3.4.2 HCLOSE.indication(uconnid,hconnid,reason,rureason,rudata)

The HCLOSE.indication service is used to inform an HFEP user that an HFEP connection is being dissolved or that a potential connection cannot be opened. The reason for the disconnection is supplied by the local HFEP entity. In the case of a remote user initiated HCLOSE.request, the remote user's reason for issuing an HCLOSE.request along with any data he may have supplied are delivered to the local HFEP user.

uconnid - an identifier assigned by the user to distinguish this connection (or potential connection) from any others.

hconnid - an identifier assigned by the local HFEP entity to distinguish this connection (or potential connection) from any others.

reason - an unsigned binary number (16 bits) specifying the reason for the disconnection.

rureason - an unsigned binary number specifying the remote user's reason for closing or refusing the connection. Rureason is limited to 16 bits. This number was specified by the remote user in his HCLOSE.request. NOTE: rureason has no meaning if reason indicates that this HCLOSE.indication was not a remote user initiated disconnection request.

rudata - a sequence of up to 32 octets of remote user supplied data in an HCLOSE.request. NOTE: rudata has no significance if reason indicates that this HCLOSE.indication was not a remote user initiated disconnection request.

3.5 HSTATUS Service

The HSTATUS service is used to determine information about the state of the local HFEP entity associated with a given connection id. The connection can be identified by either a user supplied connection identifier uconnid or a connection id (hconnid) previously supplied by an HFEP entity. The HSTATUS service is a confirmed service. Only one HSTATUS request can be outstanding at a time. HSTATUS requests are processed locally and expeditiously (i.e., as received and before subsequent HDATA

service primitives). The status returned reflects the recent state of the local HFEP connection entity.

[NOTE: At a future time the HSTATUS service might be extended to include remote HFEP status inquiries and HFEP data echoing. (This would add end-to-end significance to the HFEP service.) However, they are not included now, since these extensions are both unnecessary to HFEP operation and complex to design and implement.]

3.5.1 HSTATUS.request(uconnid,hconnid)

The HSTATUS.request primitive is used to determine the status of the local HFEP connection entity indicated either by uconnid (the user assigned connection identifier) or by hconnid (the HFEP assigned connection identifier). One, but not both of these identifiers must be specified (i.e., have a non-zero value).

uconnid - an identifier assigned by the user to distinguish this connection (or potential connection) from any others. If uconnid is zero, then hconnid is the connection identifier for which status is requested.

hconnid - an identifier assigned by the local HFEP entity to distinguish this connection (or potential connection) from any others. If hconnid is zero, then uconnid is the connection identifier for which status is requested.

3.5.2 HSTATUS.response(uconnid,hconnid,hstatus)

The HSTATUS.response primitive is issued by the HFEP entity to confirm the HSTATUS.request and to convey the status of the local HFEP connection as identified by uconnid and hconnid. The status reflects the state of the HFEP protocol entity immediately after the HSTATUS.request was issued and before the HSTATUS.response was issued.

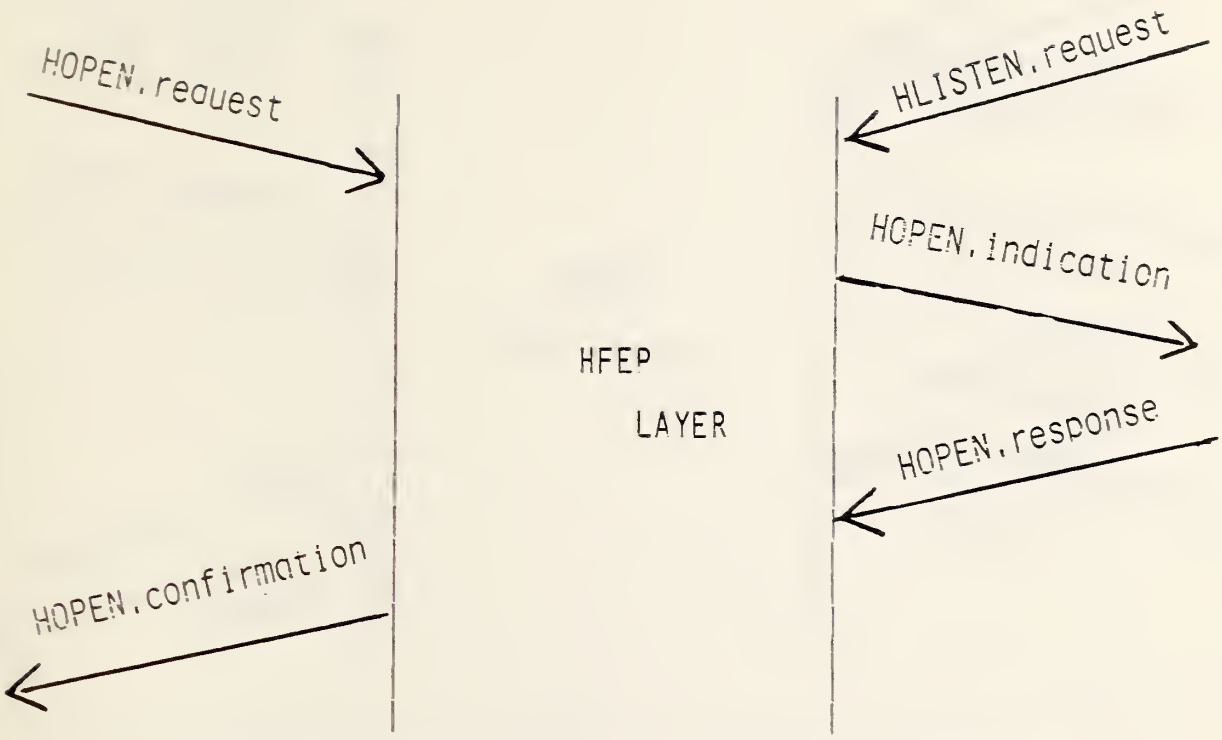
uconnid - an identifier, previously supplied to the HFEP entity to distinguish this HFEP connection (or potential connection) from any others.

hconnid - an identifier, previously supplied by the HFEP entity to distinguish this HFEP connection (or potential connection) from any others.

hstatus - a structured set of data (the exact format of which is currently unspecified) which indicate the current (or very recent) state of the local HFEP protocol machine for the HFEP connection entity identified by uconnid and hconnid. Included in hstatus are the state of the protocol machine, variables associated with the augmented state machine, and identifiers used to distinguish HFEP and lower layer connections. Also included are local IPC identifiers (to aid in possible debugging).

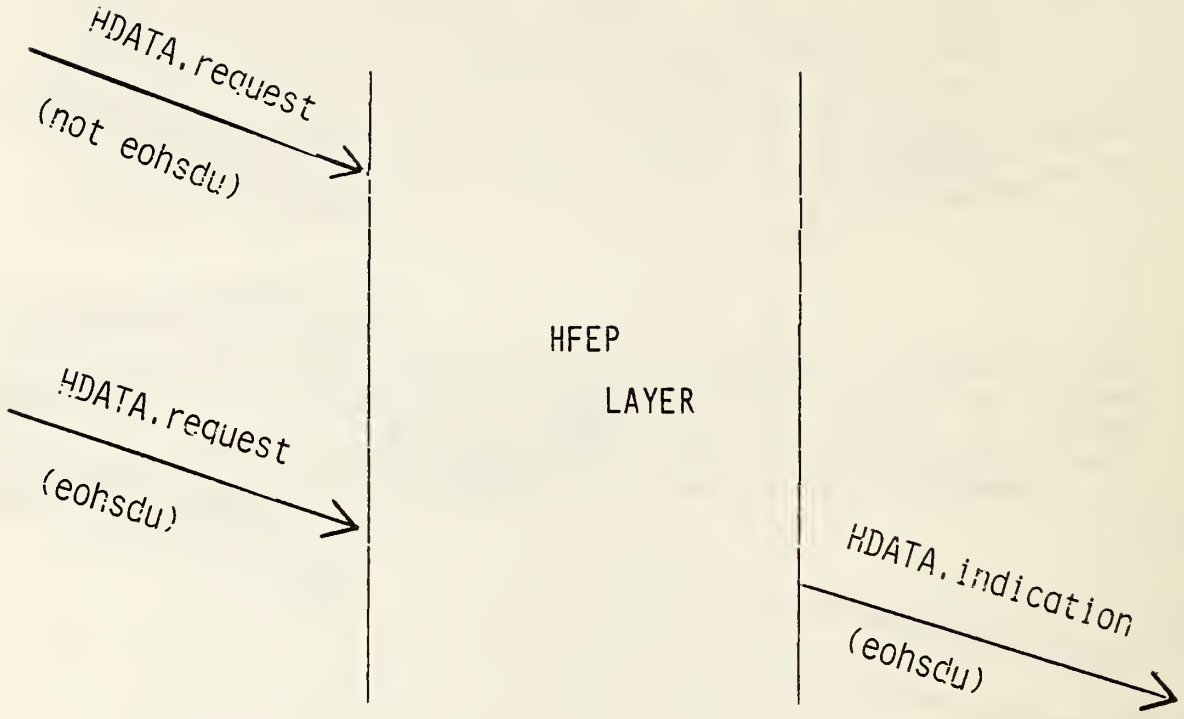
4.0 Time Sequence Diagrams

The following pages contain time sequence diagrams that illustrate the time relationships among the HFEP primitives. Time is considered to be increasing from top to bottom in these diagrams. The HFEP user interfaces for the two communicating peer entities are indicated by the vertical lines in the diagrams.



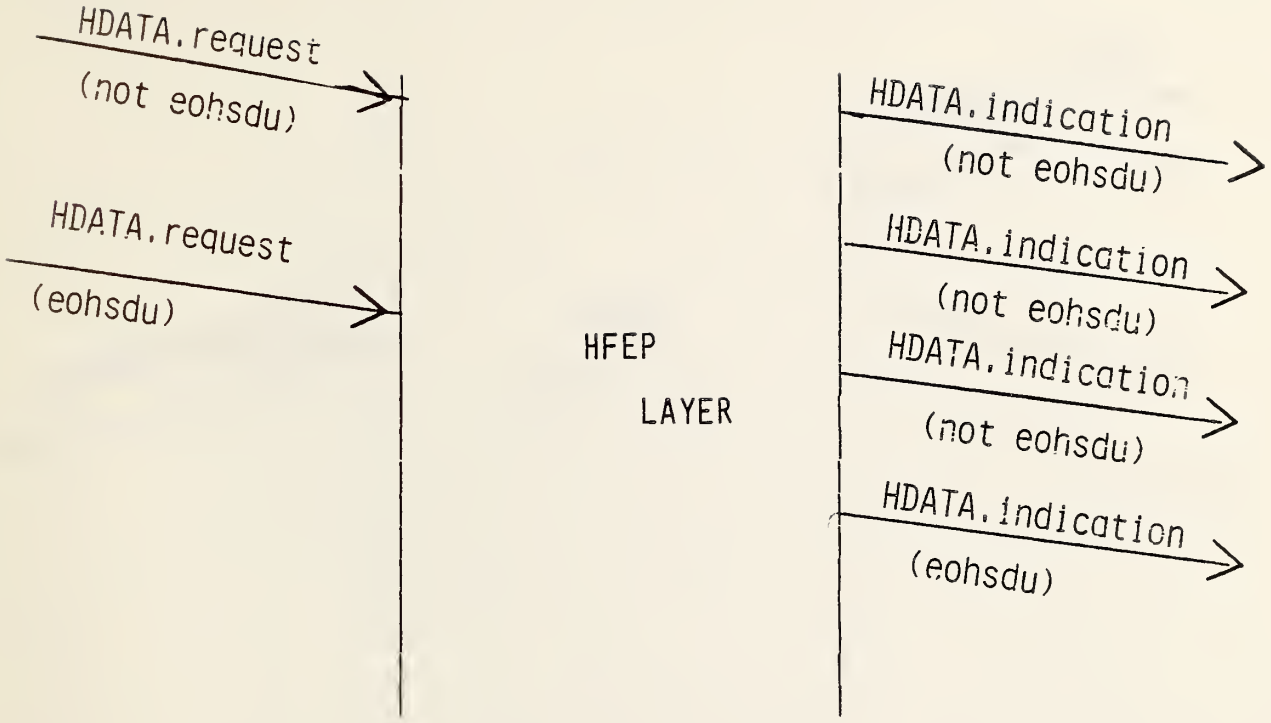
SEQUENCE OF PRIMITIVES FOR HOPEN ESTABLISHMENT

FIGURE II-1



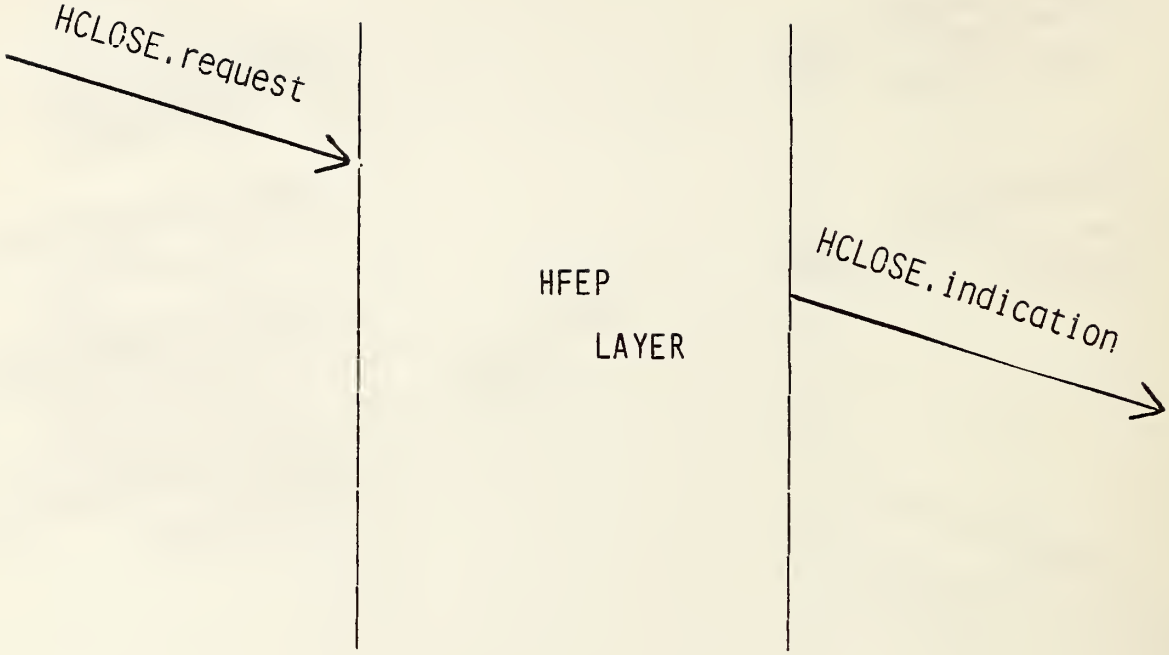
POSSIBLE SEQUENCE OF PRIMITIVES FOR HDATA SERVICE

FIGURE II-2



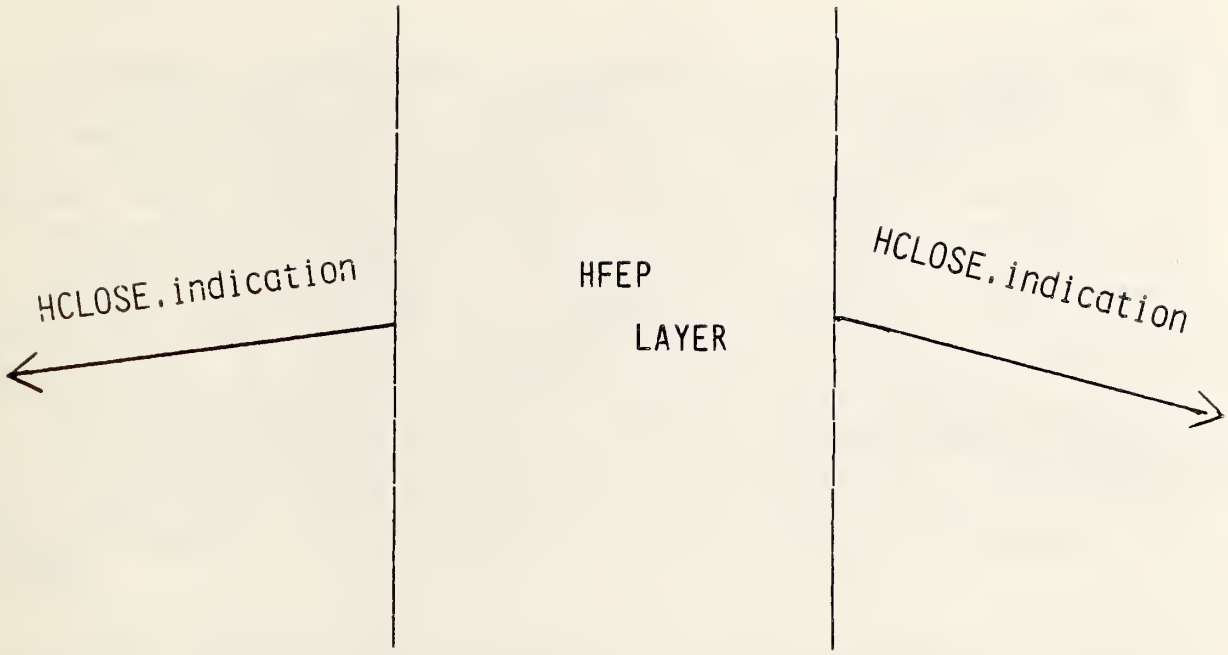
POSSIBLE SEQUENCE OF PRIMITIVES FOR HDATA SERVICE

FIGURE II-3



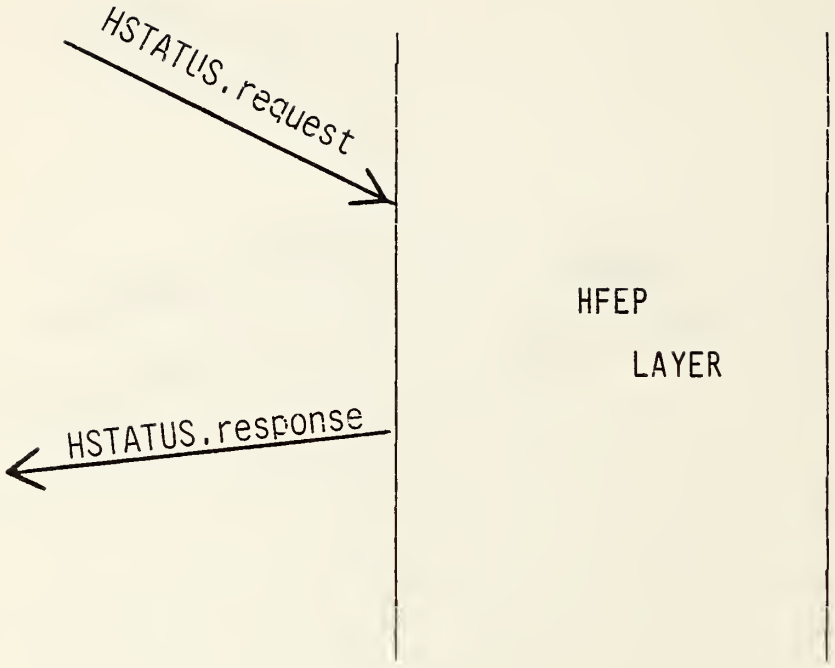
SEQUENCE OF PRIMITIVES FOR HCLOSE SERVICE

FIGURE II-4



PRIMITIVES FOR LAYER INITIATED HCLOSE SERVICE

FIGURE II-5



SEQUENCE OF PRIMITIVES FOR HSTATUS SERVICE

FIGURE II-6

5.0 Appendix A

A Solution to The N+1 Layer Rendezvous Problem

In general, the problem of rendezvous (i.e., matching peer connection requests) can be solved in several ways suggested in the ISO Reference Model. The Service Access Point Identifier (SAP-ID) of a service provider (at the N+1 Layer) can be "compiled" into a table within the N layer connection manager so that as remote connection requests arrive they can be assigned to their proper peer entity. Alternatively, remote connection requests for a given SAP-ID can be held in abeyance at the N layer until a local user requests a connection specifying that same local SAP-ID. A third, and in this author's opinion, superior method for matching peer connection requests, is to require local peer entities at the N+1 layer to register their N Layer SAP-ID's (NSAP-ID's) dynamically with the N protocol entity provider before remote connection requests arrive for that NSAP-ID. This method allows flexibility in providing services, yet appears to be straight forward to implement as well as being conservative of resources.

In this method a user at the N+1 level makes a request (N_Listen) to the N Layer to register its NSAP-ID. When a remote connection request arrives for this NSAP-ID, it is relatively straight forward for the N Layer to match the connection request to the previously registered matching NSAP-ID. The N+1 layer user who registered the NSAP-ID receives an N_Connection indication and issues an N_Connection response to indicate his acceptance (or rejection) of the connection. If accepted, the N_Connection proceeds and data transfer presumably follows to accomplish whatever service the N+1 layer is to provide. When the N+1 peer entities decide to sever the connection, one (or perhaps both) of them issue an N_Close (or N_Disconnect) request. The N_Close (or N_Disconnect) request not only severs the N_Connection but also removes the NSAP-ID entry from the N Layer's tables. Newly arriving remote connection requests for that NSAP-ID will be rejected. An N+1 layer entity must issue another N_Listen for that same NSAP-ID to allow for further connections.

To allow for multiple simultaneous connections to a given NSAP-ID, multiple N_Listens for the same NSAP-ID can be requested by one or more N+1 Layer entities. For each successful N_Listen request, the N Layer entity will make an entry into its tables indicating the NSAP-ID and the necessary corresponding Inter Process Communication (IPC) information to allow notification of the correct local process (an N+1 entity).

Arriving remote connection requests for this NSAP-ID will be assigned by the N Layer entity to any entry in its tables that a) contains the requested NSAP-ID and b) the state of any machine associated with that entry indicates that it is available for connection. The N layer entity can choose any entry which meets these criteria.

If no such entry is available, the N layer entity must transmit a PDU indicating the connection is refused. (Note: If some entry in its tables contains the requested NSAP-ID, but its state indicates that it cannot be assigned to a new connection, the N layer entity could refuse the connection but indicate as a reason "Requested Service Currently Unavailable.")

This method for matching NSAP-IDs would seem to provide all of the functionality required for current HFEP objectives. However a slight modification is useful for providing additional services and conserving resources. The N_Listen request can be modified to allow for the specification of a range of NSAP-IDs rather than just one. Such a facility is useful for "monitoring" connection requests for services that are not provided at the N+1 layer. Incoming connection requests could be noted and then refused by an N+1 layer "monitoring" entity. By allowing a single N+1 entity to accept connection requests for multiple NSAP-IDs through the use of a single entry in the N layer tables, HFEP resources (e.g., table space, buffers) can be conserved compared to resources required if each NSAP-ID needed its own individual N layer entry.

For HFEP, it is sufficient to allow (as a special case) an NSAP-ID of zero the following special meaning. If, in trying to match NSAP-ID's contained in incoming connection request PDUs with entries in its tables (as described above), an HFEP entity cannot otherwise make a match and there exists an entry with an NSAP-ID of zero, then the local process associated with that entry should be notified of the connection request.

Part III - HFEP Specification

1 Formal Definition of the HFEP Protocol Machine

1.1 Introduction

A formal specification of the HFEP is composed of nine parts: data declarations, state definitions, declarations of procedures, initialization of variables, descriptions of timers, the state transitions of the finite state machine, definitions of procedures, descriptions of primitive operations, and definition of header formats.

1.2 Data Declarations

```
/*-----------------------------------------------------------------------*/

{declarations of constants used in the specification }

const   EMPTY = empty;
        HDT_HDR_SIZE = hdt_hdr_size;
        MAX_PACK_SIZE = max_pack_size;
        NULL = null;
        NULL_DATA = null_data;
        NET_DISCON = net_discon;
        NET_RESET = net_reset;
        NET_RSTRT = net_rstrt;
        SAP_UNDEFINED = sap_undefined;
        UNDEFINED = undefined;
        USER_CLOSE = user_close;


type boolean = (        {boolean values}
        false,
        true
     ) ;

pdu_type= (     {  types of HPDU and values of HPDU.ptype }
  HOR,
```

```
  HOC,
  HDT,
  HCRI,
  HCRD
) ;

Pdu =           {All possible HPDU fields referenced in this  }
                {specification. However, not all fields are in}
 record         {any given HPDU.                              }
    ptype       : pdu_type;
    version     : int_type;
    data        : data_type;
    hlength     : int_type;
    shsap_id    : sap_type;
    dhsap_id    : sap_type;
    hreason     : int_type;
    hureason    : int_type;
    heofsduf    : boolean
  end;


machine =   {variables local to the finite state machine}
  record
    m_lhsap_id : sap_type;      {local hfep service access pt. ident.}
    m_fhsap_id : sap_type;      {foreign hfep service access pt. ident.}
    m_uconnid : int_type;       {user assigned id for this machine}
    m_hconnid : int_type;       {hfep assigned id for this machine}

    rdtlen : int_type;          {length of current receive data pdu}
    reohsdu : boolean;          {true if this pdu ends a receive hsdu}
    rbuf : buf_type;            {buffer of data from network}

    tdtlen : int_type;          {length of current transmit data pdu}
    teohsdu : boolean;          {true if this pdu ends a transmit hsdu}
    tbuf : buf_type;            {buffer of data from user}
    tmore : boolean;            {true if more data to be transmitted}
    tpdulen : int_type;         {number of octets in current}
                                {transmit packet}
    cr_udata : data_type        {user data to be delivered as part}
                                {of hopen}
  end;

                { define interface events }
```

```
interface [from U:HOPEN.request (
        uconnid : int_type;
        lhsap_id : sap_type;
        fhsap_id : sap_type;
        udata : data_type
)] ;

interface [to U:HOPEN.indication (
        uconnid : int_type;
        hconnid : int_type;
        lhsap_id : sap_type;
        fhsap_id : sap_type;
        rudata : data_type
)] ;

interface [from U:HOPEN.response (
        hconnid : int_type;
        udata : data_type
)] ;

interface [to U:HOPEN.confirm (
        uconnid : int_type;
        hconnid : int_type;
        rudata : data_type
)] ;

interface [from U:HLISTEN.request (
        uconnid : int_type;
        lhsap_id : sap_type
)] ;

interface [from U:HDATA.request (
        hconnid : int_type;
        udata : data_type;
        eohsduf : boolean
)] ;

interface [to U:HDATA.indication (
        hconnid : int_type;
        rudata : data_type;
        eohsduf : boolean
)] ;

interface [from U:HCLOSE.request (
        uconnid : int_type;
```

```
        hconnid : int_type;
        ureason : int_type;
        udata : data_type
)] ;

interface [to U:HCLOSE.indication (
        uconnid : int_type;
        hconnid : int_type;
        reason : int_type;
        rureason : int_type;
        rudata : data_type
)] ;

interface [from U:HSTATUS.request (
        uconnid : int_type;
        hconnid : int_type
)] ;

interface [to U:HSTATUS.response (
        uconnid : int_type;
        hconnid : int_type;
        hstatus : status_type
)] ;


interface [to N:NDIS.request];

interface [to N:NCON.request];

interface [to N:NINTERRUPT.request (
        Data : data_type
)] ;

interface [to N:NDATA.request (
        Data : data_type;
        Mbit : boolean
)] ;

interface [from N:NDATA.indication (
        HPDU : Pdu;
        Mbit : boolean
)] ;

interface [from N:NINTERRUPT.indication (
        HPDU : Pdu
```

```
)] ;
interface [from N:NCON.confirmation];
interface [from N:NDIS.indication];
interface [from N:NRESET.indication];
interface [from N:NRSTRT.indication];
/*-------------------------------------------------------------------*/
```

1.3 State Abbreviations

The following section defines the collections of states which will be used as state abbreviations in the "current state" field of a finite state machine transition. The last two statements of the section define the initial and final states; the finite state machine is in the initial state when it is first created, and when the final state is reached, the machine logically ceases to exist.

```
/*-------------------------------------------------------------------------*/

state any_state = (
                  HCLOSED,
                  HWFNC,
                  HWFOC,
                  HWFHRESP,
                  HOPEN,
                  HWFCRD,
                  HWFNDIS
                  );

      net_open    = (
                  HWFNC,
                  HWFOC,
                  HWFHRESP,
                  HOPEN,
                  HWFCRD
                  );

      close_ok    = (
                  HWFOC,
                  HWFHRESP,
                  HOPEN
                  );

      initial     = HCLOSED;
      final       = HCLOSED;

/*-------------------------------------------------------------------------*/
```

1.4 Procedure, Predicate, and Primitive Procedure Declarations

The following section defines the subroutines and functions used in the specification. A function returns a value; a procedure returns no value. The type of the value returned by functions is declared, as is the type of each formal parameter. All functions and procedures used in this definition are "primitive." They will be described later, but their operation will not be defined since the details are implementation dependent. Predicates are single Pascal statements which are boolean-valued and have no side effects. Predicates are used only in enabling conditions.

```
/*-------------------------------------------------------------------*/

procedure data_release (dt:data_type);                                  primitive;
procedure deliver (buf:buf_type; eohsdu:boolean);                       primitive;
procedure flush_sdu (nsdu:Pdu);                                         primitive;
procedure merge (buf:buf_type; dt:data_type; resv:int_type);            primitive;
procedure release_buf (buf:buf_type);                                   primitive;

function get_len (buf:buf_type) : int_type;                             primitive;
function buildhdr (buf:buf_type; udata_len:int_type;
          eohsdu:boolean; maxpack:int_type) : int_type;                 primitive;
function get_data (buf:buf_type; len:int_type) : data_type;             primitive;

function min (x:int_type; y:int_type) :int_type;                        primitive;

function pduHCRD (Dhreason : int_type;
                  Dureason : int_type;
                  Dudata   : data_type) : data_type;                    primitive;
function pduHOC (Cfhsap : sap_type;
                 Clhsap : sap_type;
                 Cudata  : data_type) : data_type;                      primitive;
function pduHOR (Rlhsap : sap_type;
                 Rfhsap : sap_type;
                 Rudata : data_type) : data_type;                       primitive;
function get_status : status_type;                                      primitive;
function unique_id : int_type;                                          primitive;

/*-------------------------------------------------------------------*/
```

1.5 Description of Variables

The following section defines the two initialization procedures, initialize_machine and initialize_global. These procedures assign the initial and default values to the variables used in the specification. Following the formal definition of the procedures is a list of the variables with some short text comments explaining their uses.

```
/*-------------------------------------------------------------------*/

procedure initialize_machine;
begin
    m_lhsap_id      := SAP_UNDEFINED;
    m_fhsap_id      := SAP_UNDEFINED;
    m_uconnid       := 0;
    m_hconnid       := 0;
    rdtlen          := 0;
    reohsdu         := false;
    rbuf            := EMPTY;
    tdtlen          := 0;
    teohsdu         := false;
    tbuf            := EMPTY;
    tmore           := false;
    tpdulen         := 0;
    cr_udate        := EMPTY;
end;

/*-------------------------------------------------------------------*/
```

cr_udata - User data to be delivered as part of HOPEN service.

m_fhsap_id - Foreign (HFEP peer) HFEP service access point identifier.

m_hconnid - Connection identifier assigned to the current HFEP state machine by the HFEP entity.

m_lhsap_id - Local HFEP service access point identifier.

m_uconnid - Connection identifier assigned to the current HFEP state machine by the HFEP user.

rbuf - The receive buffer. Holds data from the network before it is delivered to the user.

rdtlen - Length of current receive HFEP protocol data unit. Decremented as additional data packets arrive. Used for determining the end of an HFEP HDT.

reohsdu - Boolean variable set true if pdu currently being received is the last pdu of an HFEP service data unit.

tbuf - The transmit buffer. Holds data from the user before it is sent to the network.

tdtlen - Length of the current transmit data pdu (HDT).

teohsdu - Boolean variable set true if pdu currently being transmitted is the last pdu of an HFEP service data unit.

tmore - Boolean variable set true if more data is to be transmitted as part of the current transmit data pdu than can fit in the current X.25 packet.

tpdulen - Number of octets in the current transmit packet. (Not to be confused with the number of octets in the current transmit pdu.)

1.6 State Transitions

The following series of state transitions constitute the body of the specification -- it defines a finite state machine implementing a half-connection. Each state transition is written in the specification language which has been described earlier, and is followed by some short comments explaining the transition and its function in the protocol specification.

If the finite state machine is in some state and an input event occurs for which no transition is specified, then that input event should be ignored, i.e., no action should be taken and the finite state machine should remain in the same state.

```
/*-------------------------------------------------------------------------*/

1. ‹HCLOSED› ‹--- (1) ‹HWFNDIS› [from N:NDIS.indication]

  begin

  end;


/*-------------------------------------------------------------------------*/
```

ENABLING CONDITION:

A NDIS.indication event occurs at the Network interface; this indicates a disconnection on the Network connection.

CURRENT STATE:

The HFEP machine is in the HWFNDIS state. A Network connection is open.

ACTION:

No action is taken.

RESULTANT STATE:

The HFEP machine enters the HCLOSED state.

```
/*-------------------------------------------------------------------*/

2. ‹HCLOSED› ‹--- (1) ‹HCLOSED› [from U: HCLOSE.request]

  begin

  end;


/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An HCLOSE.request event occurs at the User interface; this indicates the user's desire to cancel an HLISTEN.request event for the HFEP service access point identifier m_lhsap_id.

CURRENT STATE:

The HFEP machine is in the HCLOSED state.

ACTION:

No action is taken.

RESULTANT STATE:

The HFEP machine remains in the HCLOSED state.
The HFEP machine disestablishes its m_lhsap_id and m_uconnid as provided by the user. (Note: In most implementations, the HFEP machine could now be deleted.)

```
/*-------------------------------------------------------------------*/

3. ‹HCLOSED› ‹--- (1) ‹HWFNDIS› [from N:NRESET.indication]

  begin
        [to N: NDIS.request]

  end;


/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

A NRESET.indication event occurs at the Network interface. This indicates an error on the Network connection.

CURRENT STATE:

The HFEP machine is in the HWFNDIS state. A Network connection is open.

ACTION:

A NDIS.request event is sent to the Network to disconnect the network connection.

RESULTANT STATE:

The HFEP machine enters the HCLOSED state.

```
/*-------------------------------------------------------------------*/
4. ‹HCLOSED› ‹--- (1) ‹HWFNDIS› [from N:NRSTRT.indication]
  begin
  end;

/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

A NRSTRT indication event occurs at the Network interface; this indicates an error on the Network connection.

CURRENT STATE:

The HFEP machine is in the HWFNDIS state. A Network connection is open.

ACTION:

No action is required.

RESULTANT STATE:

The HFEP machine enters the HCLOSED state.

```
/*-------------------------------------------------------------------*/

5. ‹HCLOSED› ‹--- (1) ‹HWFCRD› [from U: HCLOSE.request]

  begin
        [to N: NDIS.request]
  end;


/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An HCLOSE event occurs at the User interface; this indicates the user's desire to close an HFEP connection.

CURRENT STATE:

The HFEP machine is in the HWFCRD state. The peer machine has indicated that it wishes to close the HFEP connection and the local HFEP machine is awaiting the reason for the close.

ACTION:

The Network disconnect request is made to terminate the connection.

RESULTANT STATE:

The HFEP machine enters the HCLOSED state.

```
/*-----------------------------------------------------------------*/

6. ‹HCLOSED› ‹--- (1) ‹net_open› [from N:NDIS.indication]

  begin
        [to U: HCLOSE.indication (
                        uconnid := m_uconnid,
                        hconnid := m_hconnid,
                        reason := NET_DISCON,
                        rureason := NULL,
                        rudata := NULL_DATA)]

  end;


/*-----------------------------------------------------------------*/
```

ENABLING CONDITION:

A NDIS.indication event occurs at the Network interface; this indicates an HFEP protocol error or Network error, because the NDIS indication was not expected from this state.

CURRENT STATE:

The HFEP machine is neither in the HWFNDIS nor in the HCLOSED state. A Network connection is open.

ACTION:

The HFEP machine informs the user via an HCLOSE.indication event that the HFEP connection is being terminated.

RESULTANT STATE:

The HFEP machine enters the HCLOSED state.

```
/*-------------------------------------------------------------------*/

7. ‹HWFCRD› ‹--- (1) ‹HWFCRD› [from N: NDATA.indication]
                                ((rdtlen = 0)  and
                                ([from N:HPDU.ptype] = HDT))

  begin
        rdtlen := [from N:HPDU.hlength];
        flush_sdu([from N:HPDU]);
        if [from N:Mbit] then
            begin

                rdtlen := rdtlen - MAX_PACK_SIZE + HDT_HDR_SIZE;

            end
        else
            begin
                rdtlen := 0;
            end
  end;

/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An NDATA event occurs at the Network interface; this indicates the arrival of some Network data. The HFEP machine is not yet processing an HFEP HDT pdu since rdtlen equals zero, but a new HDT has arrived.

CURRENT STATE:

The HFEP machine is in the HWFCRD state; therefore it is prepared to receive data and discard it until an HCRD pdu arrives.

ACTION:

The length of the current HFEP HDT pdu is extracted and placed in rdtlen. The X.25 packet can then be discarded using primitive procedure flush_sdu. If more data is to follow as part of the current HFEP HDT, the amount of remaining data is recomputed and stored into rdtlen. Otherwise, rdtlen is reset to zero in anticipation of either the next HDT to arrive or an HCRD.

RESULTANT STATE:

The HFEP machine remains in the HWFCRD state.

```
/*-------------------------------------------------------------------*/

8. ‹HWFCRD› ‹--- (1) ‹HWFCRD› [from N: NDATA.indication]
                                (rdtlen › 0)

  begin
        flush_sdu([from N:HPDU.data]);
        if [from N:Mbit] then
            begin
                rdtlen := rdtlen - MAX_PACK_SIZE;
            end
        else
            begin
                rdtlen := 0;
            end
  end;

/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An NDATA event occurs at the Network interface; this indicates the arrival of some Network data. The HFEP machine has already started processing an HFEP HDT pdu since rdtlen is not equal to zero.

CURRENT STATE:

The HFEP machine is in the HWFCRD state; therefore it is prepared to receive data.

ACTION:

The X.25 packet is discarded using primitive procedure flush_sdu. If more data is to follow as part of the current HFEP HDT, the amount of remaining data is recomputed and stored into rdtlen. Otherwise, rdtlen is reset to zero in anticipation of either the next HDT to arrive or an HCRD.

RESULTANT STATE:

The HFEP machine remains in the HWFCRD state.

```
/*-------------------------------------------------------------------*/

9. ‹HWFNDIS› ‹--- (1) ‹HWFNDIS› [from N: NDATA.indication]

  begin
        flush_sdu([from N:HPDU])
  end;

/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An NDATA event occurs at the Network interface; this indicates the arrival of some Network sdu.

CURRENT STATE:

The HFEP machine is in the HWFNDIS state; therefore it is awaiting a Network disconnect event.

ACTION:

The Network sdu (data) is discarded using primitive procedure flush_sdu.

RESULTANT STATE:

The HFEP machine remains in the HWFNDIS state.

```
/*-------------------------------------------------------------------*/

10. ‹HCLOSED› ‹--- (1) ‹HWFCRD› [from N: NDATA.indication]
                                ((rdtlen = 0)  and
                                ([from N:HPDU.ptype] = HCRD))

  begin
        [to U: HCLOSE.indication (
                          uconnid := m_uconnid,
                          hconnid := m_hconnid,
                          reason := USER_CLOSE,
                          rureason := [from N:HPDU.hureason],
                          rudata := [from N:HPDU.data])];
        [to N: NDIS.request]

  end;

/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An NDATA event occurs at the Network interface; this indicates the arrival of some Network data. The HFEP machine is not yet processing an HFEP HDT pdu since rdtlen equals zero, but a HCRD has arrived.

CURRENT STATE:

The HFEP machine is in the HWFCRD state; therefore it is prepared to receive data and discard it until an HCRD pdu arrives.

ACTION:

The HFEP machine informs the user via an HCLOSE.indication event that the HFEP connection is being terminated. A NDIS.request event is sent to the Network to disconnect the network connection.

RESULTANT STATE:

The HFEP machine enters the HCLOSED state.

```
/*-------------------------------------------------------------------*/

11. ‹HOPEN› ‹--- (1) ‹HOPEN› [from N: NDATA.indication]
                              ((rdtlen = 0)  and
                              ([from N:HPDU.ptype] = HDT))

  begin
        merge(rbuf,[from N:HPDU.data],NULL);
        rdtlen := [from N:HPDU.hlength];
        reohsdu := [from N:HPDU.heofsduf];
        if [from N:Mbit] then
            begin

                rdtlen := rdtlen - MAX_PACK_SIZE + HDT_HDR_SIZE;

            end
        else
            begin
                deliver(rbuf,reohsdu);
                release_buf(rbuf);
                rdtlen := 0;
            end
  end;

/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An NDATA event occurs at the Network interface; this indicates the arrival of some Network data. The HFEP machine is not yet processing an HFEP HDT pdu since rdtlen equals zero, but a new HDT has arrived.

CURRENT STATE:

The HFEP machine is in the HOPEN state; therefore it is prepared to receive data.

ACTION:

The Network data is merged into the receive buffer. If more data is to follow as part of the current HFEP HDT, the amount of remaining data is recomputed and stored into rdtlen. Otherwise, the data received is

delivered to the user using primitive procedure deliver and rdtlen is reset to zero in anticipation of the next HDT to arrive.

RESULTANT STATE:

The HFEP machine remains in the HOPEN state.

```
/*-------------------------------------------------------------------*/

12. ‹HOPEN› ‹--- (1) ‹HOPEN› [from N: NDATA.indication]
                              (rdtlen › 0)

  begin
        merge(rbuf,[from N:HPDU.data],NULL);
        if [from N:Mbit] then
            begin
                rdtlen := rdtlen - MAX_PACK_SIZE;
            end
        else
            begin
                deliver(rbuf,reohsdu);
                release_buf(rbuf);
                rdtlen := 0;
            end
  end;

/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An NDATA event occurs at the Network interface; this indicates the arrival of some Network data. The HFEP machine has already started processing an HFEP HDT pdu since rdtlen is not equal to zero.

CURRENT STATE:

The HFEP machine is in the HOPEN state; therefore it is prepared to receive data.

ACTION:

The Network data is merged into the receive buffer. If more data is to follow as part of the current HFEP HDT, the amount of remaining data is recomputed and stored into rdtlen. Otherwise, the data in the receive buffer is delivered to the user using primitive procedure deliver and rdtlen is reset to zero in anticipation of the next HDT to arrive.

RESULTANT STATE:

The HFEP machine remains in the HOPEN state.

```
/*-----------------------------------------------------------------*/

13. ‹HOPEN› ‹--- (1) ‹HOPEN› [from U: HDATA.request]

  begin
        merge(tbuf,[from U:udata],HDT_HDR_SIZE);
        tdtlen := get_len(tbuf);
        teohsdu := [from U:eohsduf];
        if (teohsdu = true) or (tdtlen ›= MAX_PACK_SIZE - HDT_HDR_SIZE)then
            begin
                tpdulen := buildhdr(tbuf,tdtlen,teohsdu,MAX_PACK_SIZE);
                if tdtlen › tpdulen - HDT_HDR_SIZE
                    then tmore := true
                    else tmore := false;

                [to N: NDATA.request
                    (Data := get_data(tbuf,tpdulen),
                     Mbit := tmore)];

                tdtlen := tdtlen - tpdulen + HDT_HDR_SIZE;

                while tdtlen › 0 do
                    begin
                        tpdulen := min(tdtlen,MAX_PACK_SIZE);

                        if tdtlen › tpdulen
                            then tmore := true
                            else tmore := false;

                        [to N: NDATA.request
                            (Data := get_data(tbuf,tpdulen),
                             Mbit := tmore)];

                        tdtlen := tdtlen - tpdulen;
                    end
            end
  end;

/*-----------------------------------------------------------------*/
```

ENABLING CONDITION:

A HDATA event occurs at the User interface; this indicates the arrival of some user data.

CURRENT STATE:

The HFEP machine is in the HOPEN state; therefore it is prepared to receive data.

ACTION:

The User data are merged into the transmit buffer. Space is reserved in the buffer for an HDT header of size HDT_HDR_SIZE. The length of the data is derived by primitive function get_len; The variable teohsdu is set true if the user has indicated that these data are the end of an HFEP service data unit. If the user has not indicated the end of an HFEP service data unit and there are not enough data to fill a packet (after allowing for the HDT header) then no further action is performed. The data are held in anticipation of further data to complete the service data unit.

If either the user has indicated end of HSDU or there are enough data to fill a packet, then primitive function buildhdr is called to construct an HDT header. The packet is sent by an NDATA request. If there are further data to be sent, the Mbit is set in the packet. Additional packets of maximum size MAX_PACK_SIZE are sent until there are no data left to be sent.

RESULTANT STATE:

The HFEP machine remains in the HOPEN state.

```
/*-------------------------------------------------------------------*/

14. ‹HWFCRD› ‹--- (1) ‹HWFCRD› [from U: HDATA.request]

  begin
        data_release([from U:Data])
  end;


/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An HDATA.request event occurs at the User interface; this indicates the user's desire to send data.

CURRENT STATE:

The HFEP machine is in the HWFCRD state, has received an HCRI pdu indicating the peer machine's desire to close, and is awaiting an HCRD pdu containing the reason for the close.

ACTION:

The HFEP machine invokes the primitive procedure data_release to discard the user's data since the other side is closing. [Note: As a local implementation matter, the user may be informed via an HSTATUS.indication that the data is discarded because his peer is closing.]

RESULTANT STATE:

The HFEP machine remains in the HWFCRD state.

```
/*-------------------------------------------------------------------*/

15. ‹HCLOSED› ‹--- (1) ‹HCLOSED› [from U: HLISTEN.request]

  begin
        m_uconnid := [from U:uconnid];
        m_hconnid := unique_id;
        m_lhsap_id := [from U:lhsap_id]
  end;


/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An HDATA.request event occurs at the User interface; this indicates the user's desire to await an HOPEN.indication event for the HFEP service access point identifier lhsap_id.

CURRENT STATE:

The HFEP machine is in the HCLOSED state.

ACTION:

The HFEP machine establishes its m_lhsap_id and m_uconnid as provided by the user. A unique hconnid, provided by primitive function unique_id, is assigned to the machine.

RESULTANT STATE:

The HFEP machine remains in the HCLOSED state.

```
/*-------------------------------------------------------------------*/

16. ‹HWFNC› ‹--- (1) ‹HCLOSED› [from U: HOPEN.request]

  begin
        m_uconnid := [from U:uconnid];
        m_hconnid := unique_id;
        m_lhsap_id := [from U:lhsap_id];
        m_fhsap_id := [from U:fhsap_id];
        cr_udata := [from U:udata];
        [to N: NCON.request]
  end;


/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An HOPEN.request event occurs at the User interface; this indicates the user's desire to open an HFEP connection from local SAP_identifier lhsap_id to the peer SAP_identifier fhsap_id.

CURRENT STATE:

The HFEP machine is in the HCLOSED state.

ACTION:

The user parameters are stored in the state machine and the primitive procedure unique_id is called to generate unique identifier for this machine. Finally, a network connection request is made to provide a network connection on which to build an HFEP connection.

RESULTANT STATE:

The HFEP machine enters the HWFNC state.

```
/*-------------------------------------------------------------------*/

17. ‹HCLOSED› ‹--- (1) ‹net_open› [from N:NRESET.indication]

  begin
        [to U: HCLOSE.indication (
                        uconnid := m_uconnid,
                        hconnid := m_hconnid,
                        reason := NET_RESET,
                        rureason := NULL,
                        rudata := NULL_DATA)];
        [to N: NDIS.request]

  end;


/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

A NRESET.indication event occurs at the Network interface; this indicates an error on the Network connection.

CURRENT STATE:

The HFEP machine is neither in the HWFNDIS nor in the HCLOSED state. A Network connection is open.

ACTION:

The HFEP machine informs the user via an HCLOSE.indication event that the HFEP connection is being terminated. A NDIS.request event is sent to the Network to disconnect the network connection.

RESULTANT STATE:

The HFEP machine enters the HCLOSED state.

```
/*-------------------------------------------------------------------*/

18. ‹HCLOSED› ‹--- (1) ‹net_open› [from N:NRSTRT.indication]

  begin
       [to U: HCLOSE.indication (
                        uconnid := m_uconnid,
                        hconnid := m_hconnid,
                        reason := NET_RSTRT,
                        rureason := NULL,
                        rudata := NULL_DATA)]
  end;


/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

A NRSTRT.indication event occurs at the Network interface; this indicates an error on the Network connection.

CURRENT STATE:

The HFEP machine is neither in the HWFNDIS nor in the HCLOSED state. A network connection is open.

ACTION:

The HFEP machine informs the user via an HCLOSE.indication event that the HFEP connection is being terminated. The NRSRT.indication event indicates that the Network connection has been broken.

RESULTANT STATE:

The HFEP machine enters the HCLOSED state.

```
/*-------------------------------------------------------------------*/

19. ‹HOPEN› ‹--- (1) ‹HWFHRESP› [from U: HOPEN.response]

  begin
        [to N: NDATA.request
            (Data := pduHOC(
                        m_fhsap_id,
                        m_lhsap_id,
                        [from U:udata]
                        ),
             Mbit := false)]

  end;
```

```
/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An HOPEN.response event occurs at the User interface. This indicates that the user wishes to complete the opening of an HFEP connection.

CURRENT STATE:

The HFEP machine is in the HWFHRESP state.

ACTION:

The HFEP machine invokes the primitive function pduHOC to build an HOC HFEP protocol data unit, which includes the user data. The pdu is sent to the peer HFEP machine using an NDATA request.

RESULTANT STATE:

The HFEP machine enters the HOPEN state.

```
/*-------------------------------------------------------------------*/

20. ‹same_state› ‹--- (1) ‹any_state› [from U: HSTATUS.request]

  begin
        [to U: HSTATUS.response
            ( uconnid := m_uconnid,
              hconnid := m_hconnid,
              hstatus := get_status)]

  end;

/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An HSTATUS.request event occurs at the User interface; this indicates the user's desire to determine the status of the HFEP machine.

CURRENT STATE:

The HFEP machine is in any state.

ACTION:

The User and HFEP assigned connection identifiers are copied to the User Interface and the HFEP machine invokes the primitive function get_status to determine and return current status to the user.

RESULTANT STATE:

The HFEP machine remains in its current state.

```
/*-------------------------------------------------------------------*/

21. ‹HWFNDIS› ‹--- (1) ‹close_ok› [from U: HCLOSE.request]

  begin
        release_buf(rbuf);
        [to N: NINTERRUPT.request (Data := HCRI)];
        [to N: NDATA.request (Data := pduHCRD(
                                        USER_CLOSE,
                                        [from U: ureason],
                                        [from U: udata]
                                        ),
                              Mbit := false)
        ]
  end;
```

```
/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An HCLOSE.request event occurs at the User interface; this indicates the user's desire to close an HFEP connection.

CURRENT STATE:

The HFEP machine is in one of the close_ok states. A network connection exists and the other HFEP entity must be informed and the user data delivered.

ACTION:

The primitive procedure release_buf is called to discard any data stored in the receive buffer. An interrupt (expedited data one octet long) is sent to the peer machine to inform it that it may start discarding data in anticipation of an HCRD, which includes the user's reason for the closing of the connection. The HCRD is then sent.

RESULTANT STATE:

The HFEP machine enters the HWFNDIS state.

```
/*-------------------------------------------------------------------*/

22. ‹HWFCRD› ‹--- (1) ‹close_ok› [from N: NINTERRUPT.indication]
                                    ([from N:HPDU.ptype] = HCRI)

  begin
        release_buf(rbuf)
  end;


/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An NINTERRUPT event occurs at the Network interface, indicating the arrival of some Network interrupt data containing an HFEP HCRI pdu. The HCRI indicates the peer HFEP's desire to close the connection.

CURRENT STATE:

The HFEP machine is in one of the close_ok states. A network connection exists.

ACTION:

The primitive procedure release_buf is called to discard any data stored in the receive buffer. [Note: As a local implementation matter, an HSTATUS.indication may be issued to inform the user that an HCLOSE is expected.]

RESULTANT STATE:

The HFEP machine enters the HWFCRD state in anticipation of an HCRD pdu, which includes the peer entity's reason for the closing of the connection.

```
/*-------------------------------------------------------------------*/

23. ‹HCLOSED› ‹--- (1) ‹HWFNDIS› [from N: NINTERRUPT.indication]
                                    ([from N:HPDU.ptype] = HCRI)

  begin

        [to N: NDIS.request]

  end;


/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An NINTERRUPT event occurs at the Network interface, indicating the arrival of some Network interrupt data containing an HFEP HCRI pdu. The HCRI indicates the peer HFEP's desire to close the connection.

CURRENT STATE:

The HFEP machine is in the HWFNDIS state. It has sent out an HCRI to indicate to the peer machine that it wishes to close the connection.

ACTION:

A NDIS.request event is sent to the Network to disconnect the network connection.

RESULTANT STATE:

The HFEP machine enters the HCLOSED state.

```
/*-------------------------------------------------------------------*/

24. ‹HWFOC› ‹--- (1) ‹HWFNC› [from N: NCON.confirmation]

  begin

        [to N: NDATA.request
            (Data := pduHOR(
                        m_lhsap_id,
                        m_fhsap_id,
                        cr_udata
                        ),
             Mbit := false)]

  end;

/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An NCON confirmation event occurs at the Network interface to indicate that a network connection is now open.

CURRENT STATE:

The HFEP machine is in the HWFNC state. It has been awaiting a network connection.

ACTION:

Primitive function pduHOR is invoked to construct an HOR HFEP pdu to convey to the peer machine the desire to open an HFEP connection. An NDATA.request event containing the HOR is sent via a Network data request.

RESULTANT STATE:

The HFEP machine enters the HCLOSED state.

```
/*-------------------------------------------------------------------*/

25. ‹HOPEN› ‹--- (1) ‹HWFOC› [from N: NDATA.indication]
                                ((rdtlen = 0) and
                                ([from N:HPDU.ptype] = HOC))

  begin

        [to U: HOPEN.confirm (
                        uconnid := m_uconnid,
                        hconnid := m_hconnid,
                        rudata := [from N:HPDU.data]
                        )]

  end;


/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An NDATA event occurs at the network interface indicating the arrival of an HOC pdu.

CURRENT STATE:

The HFEP machine is in the HWFOC state.

ACTION:

The HFEP machine informs the user via an HOPEN.confirmation event that the HFEP connection is now established.

RESULTANT STATE:

The HFEP machine enters the HOPEN state.

```
/*-------------------------------------------------------------------*/

26. ‹HWFHRESP› ‹--- (1) ‹HCLOSED› [from N: NDATA.indication]
                                    ((rdtlen = 0) and
                                    ([from N:HPDU.ptype] = HOR))

  begin

        m_lhsap_id := [from N:HPDU.dhsap_id];
        m_fhsap_id := [from N:HPDU.shsap_id];
        [to U: HOPEN.indication (
                        uconnid := m_uconnid,
                        hconnid := m_hconnid,
                        lhsap_id := m_lhsap_id,
                        fhsap_id := m_fhsap_id,
                        rudata := [from N:HPDU.data]
                )]


  end;

/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An NDATA event occurs at the Network interface. This indicates the arrival of some Network data containing an HFEP HOR pdu.

CURRENT STATE:

The HFEP machine is in the HCLOSED state. It has been awaiting the HOR after the user had performed an HLISTEN.request.

ACTION:

An HOPEN.indication user event is sent to the user interface, after the destination and source sap_id's are copied to the local state machine as m_lhsap_id and m_fhsap_id, respectively. The lhsap_id must be copied (although set as part of the HLISTEN.request transaction) for the case in which it was originally specified as zero, i.e., willing to accept connections for any local sap_id.

RESULTANT STATE:

The HFEP machine enters the HWFHRESP state.

```
/*-------------------------------------------------------------------*/

27. ‹HCLOSED› ‹--- (1) ‹HWFNC› [from U: HCLOSE.request]

  begin
        [to N: NDIS.request]
  end;


/*-------------------------------------------------------------------*/
```

ENABLING CONDITION:

An HCLOSE event occurs at the User interface; this indicates the user's desire to close an HFEP connection.

CURRENT STATE:

The HFEP machine is in the HWFNC state awaiting the opening of a network connection.

ACTION:

The Network disconnect request is made to terminate the connection.

RESULTANT STATE:

The HFEP machine enters the HCLOSED state.

1.7 Primitive Procedures and Functions

```
/*-----------------------------------------------------------------*/
```

The following describe the effects of primitive functions and procedures used in the specification. The exact nature of the operations remain undefined since they depend on the nature of the implementation.

```
procedure data_release (dt:data_type);
```

This primitive discards any data associated with dt. All storage for dt can be reused as the data are not needed.

```
procedure deliver (buf:buf_type; eohsdu:boolean);
```

This primitive, using a [to U:HDATA.indication] interface event, delivers any data associated with buf to the user. The User is informed that an end of HFEP service data unit has occured if eohsdu is true.

```
procedure flush_sdu (nsdu:Pdu);
```

This primitive discards the Network service data unit associated with nsdu. All resources (primarily storage) associated with nsdu can be reused as they are no longer needed.

```
procedure merge (buf:buf_type; dt:data_type; resv:int_type);
```

The merge primitive removes all data from dt and places (concatenates) it at the the end of buf after reserving space for resv octets of data. This space is normally reserved for header information in a transmited HDATA pdu.

```
procedure release_buf (buf:buf_type);
```

Primitive procedure release_buf frees all resources (primarily storage for data) associated with buffer buf. These resources are no longer needed.

```
function get_len (buf:buf_type) : int_type;
```

The primitive function get_len returns the length in octets of the data stored in buffer buf. This length does NOT include any reserved space for header octets.

```
function buildhdr (buf:buf_type; udata_len:int_type; eohsdu:boolean;
                   maxpack:int_type) : int_type;
```

Primitive function buildhdr builds an HFEP data pdu (HDT) header at the beginning of buffer buf. The HDT includes udata_len octets of data. The end of HFEP service data unit flag is set in the header if eohsdu is true. The maximum packet size that can be sent is indicated by maxpack. Buildhdr returns the number of octets (both HDT header and user data) in the packet being built.

```
function get_data (buf:buf_type; len:int_type) : data_type;
```

Primitive function get_data returns the first len octets of data (possibly including user data as well as header) from buffer buf. The data returned is removed from buf.

```
primitive min (x:int_type, y:int_type);
```

The min primitive returns the lesser of its two arguments.

```
function pduHCRD (Dhreason  : int_type;
                  Dureason  : int_type;
                  Dudata    : data_type) : data_type;
```

Primitive function pduHCRD builds an HFEP HCLOSE request (HCRD) pdu which is returned as its value. The reason Dhreason and user specified reason Dureason as well as the user supplied data Dudata are placed into the HCRD.

```
function pduHOC (Cfhsap : sap_type;
                 Clhsap : sap_type;
                 Cudata : data_type) : data_type;
```

Primitive function pduHOC builds an HFEP HOPEN confirm (HOC) pdu which

is returned as its value. The foreign and local service access points, Cfhsap and Clhsap respectively, and user supplied data Cudata are placed into the HOC. Cfhsap is placed into the source HFEP SAP (SHSAP-ID) of the HOC and Clhsap is placed into the destination HFEP SAP (DHSAP-ID).

```
function pduHOR (Rlhsap : sap_type;
                 Rfhsap : sap_type;
                 Rudata : data_type) : data_type;
```

Primitive function pduHOR builds an HFEP HOPEN request (HOR) pdu which is returned as its value. The local and foreign service access points, Rlhsap and Rfhsap respectively, and user supplied data Rudata are placed into the HOR. Rlhsap is placed into the source HFEP SAP (SHSAP-ID) of the HOR and Rfhsap is placed into the destination HFEP SAP (DHSAP-ID).

```
function get_status : status_type;
```

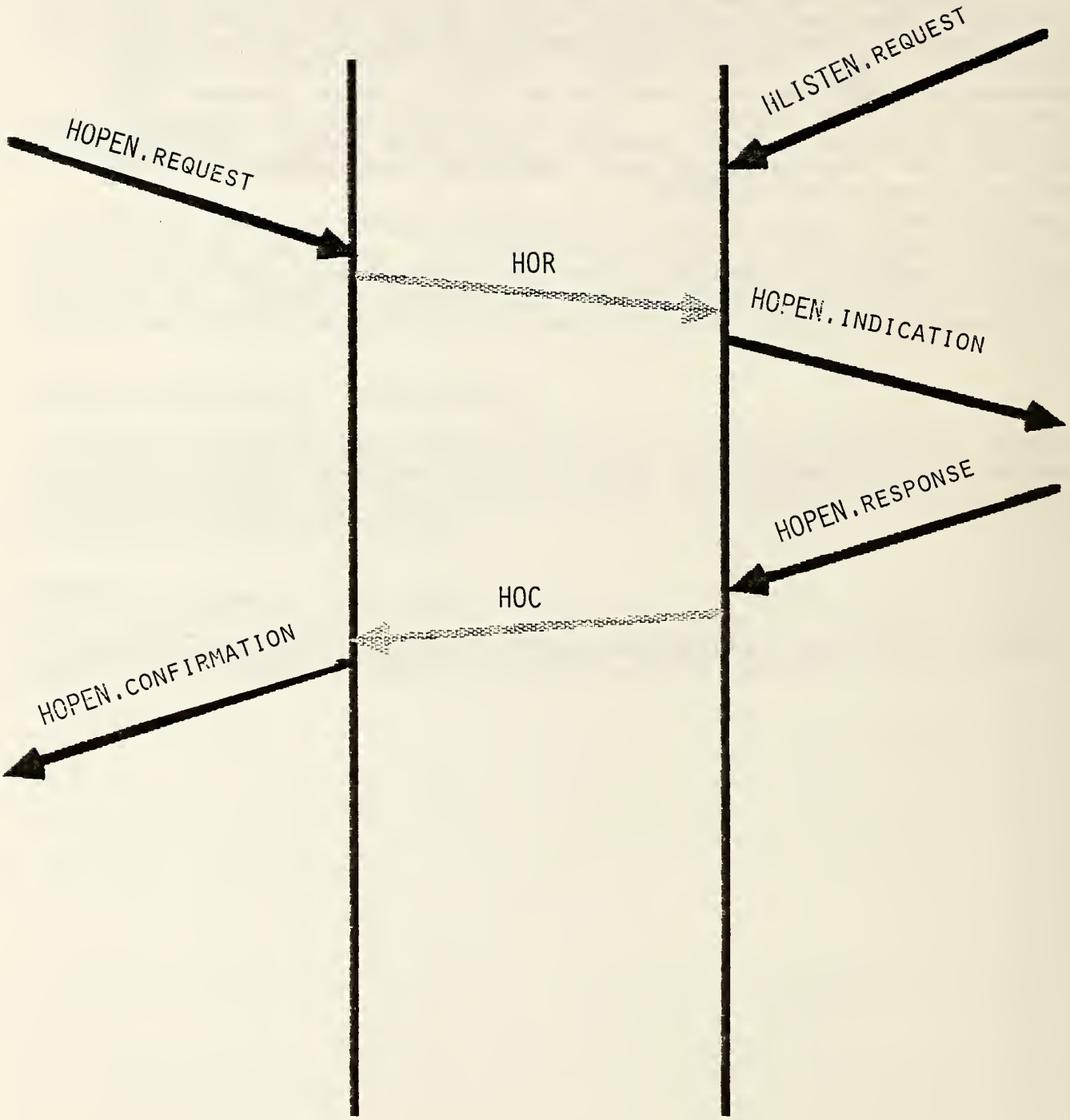
Primitive function get_status returns the status of local HFEP machine. Status information returned should include, at minimum, all the scalar information within the HFEP machine record, plus additional information on the data types and buffers.

```
function unique_id : int_type;
```

Primitive function unique_id returns an hconnid that is not in use. This value is used to identify an HFEP machine uniquely.

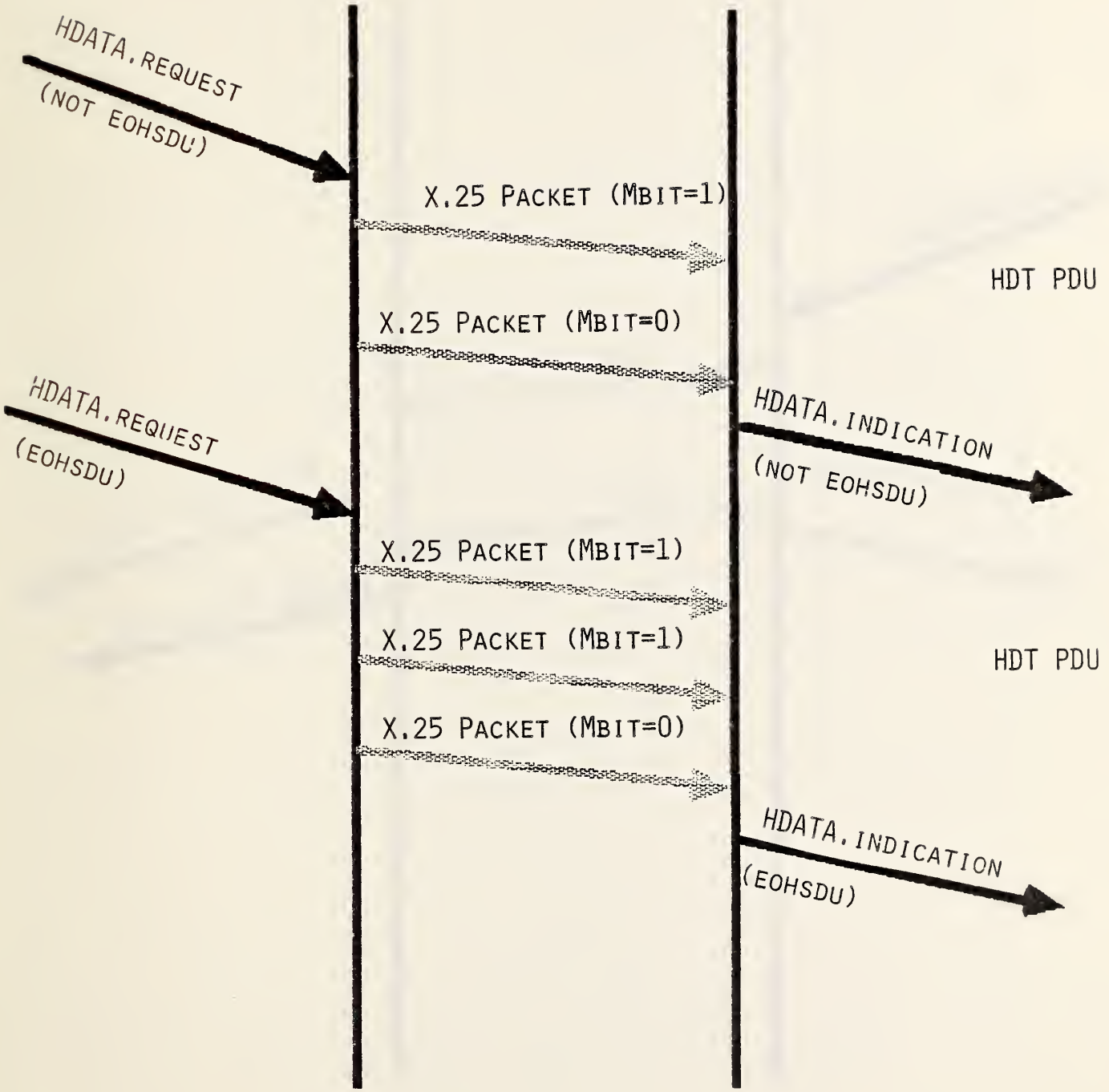
2.0 Time Sequence Diagrams

Figures III-1 through III-5 are Time Sequence Diagrams that illustrate the time relationships among the HFEP primitives. The solid arrows on the sides of the diagrams represent the primitives themselves. The shaded arrows between the vertical lines represent the HFEP protocol data units (PDU's) sent between the communicating peer entities. Time is considered to be increasing from top to bottom in these diagrams. The HFEP user interfaces for the two communicating peer entities are indicated by the vertical lines in the diagrams.



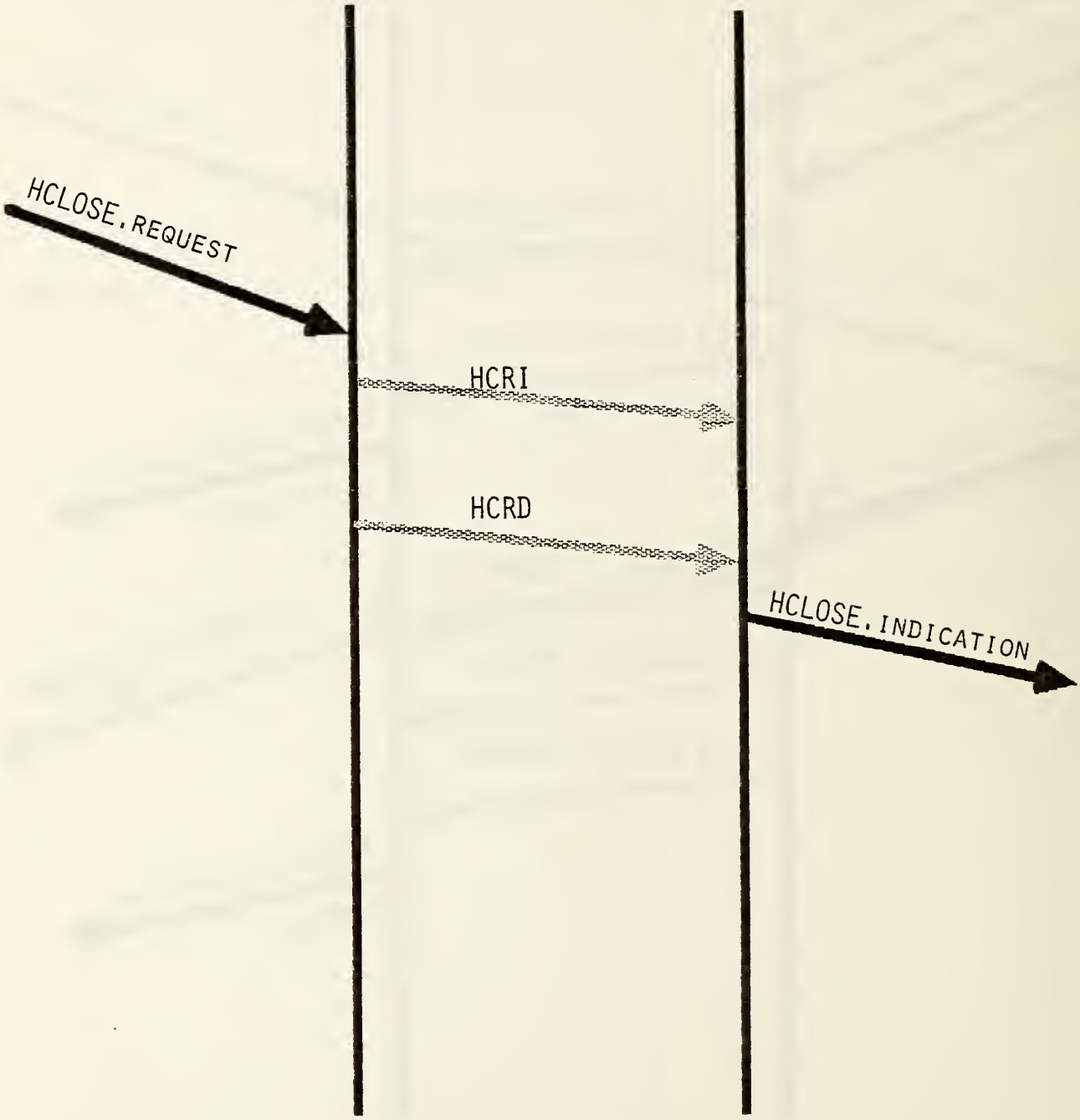
HFEP HOPEN ESTABLISHMENT

FIGURE III-1



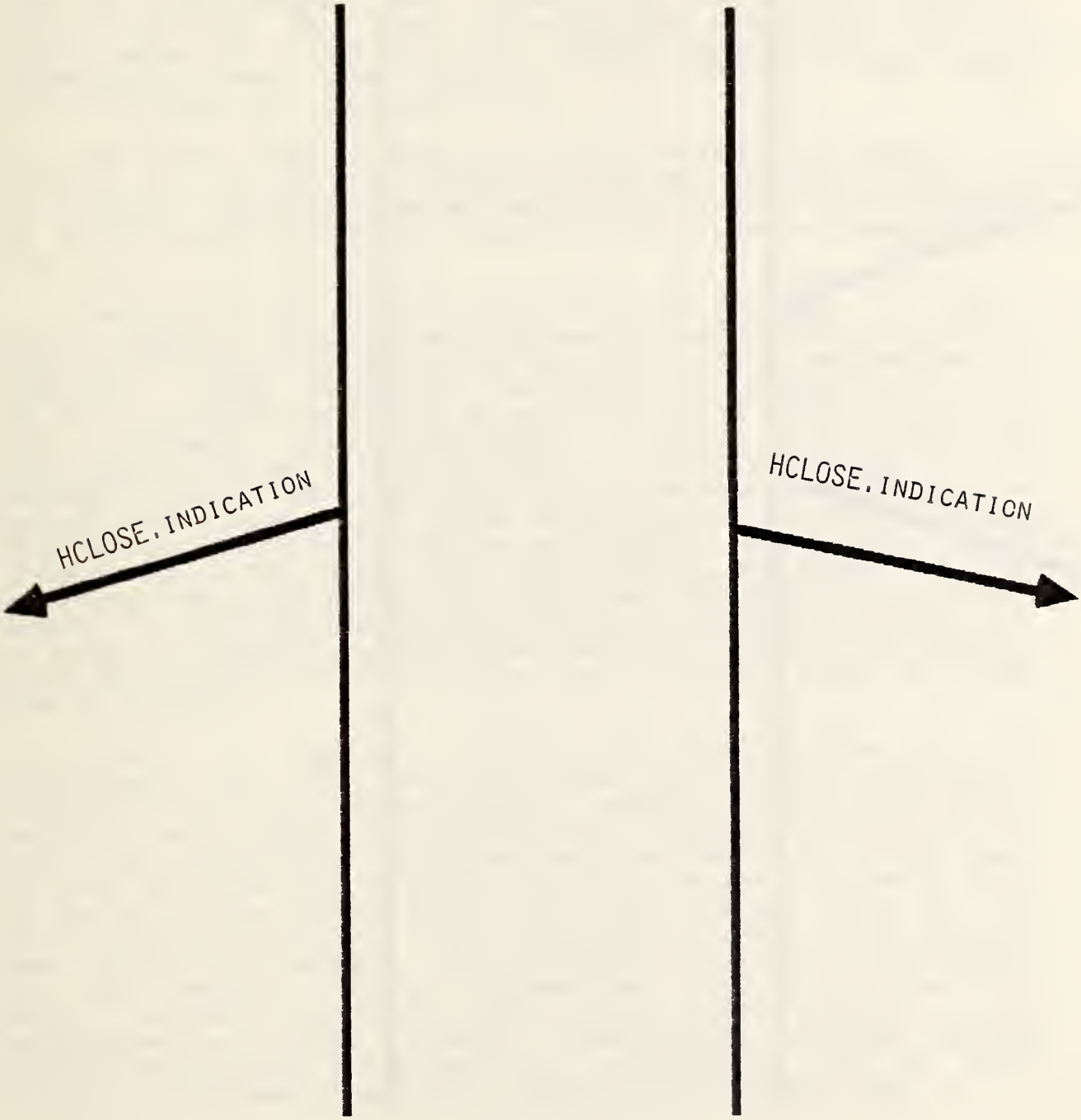
POSSIBLE SEQUENCE FOR HFEP HDATA SERVICE

FIGURE III-2



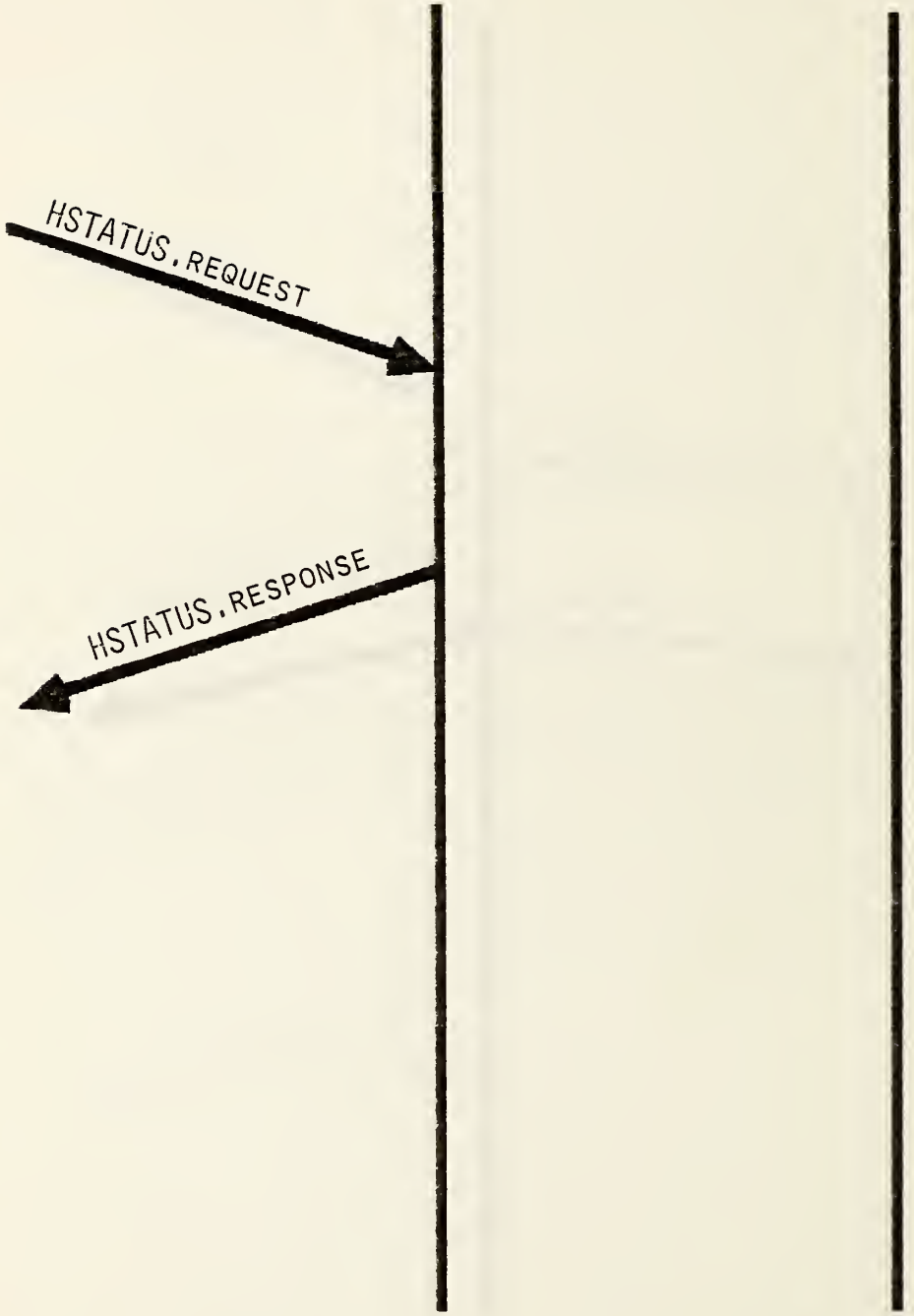
NORMAL HFEP HCLOSE SERVICE

FIGURE III-3



LAYER INITIATED HCLOSE SERVICE

FIGURE III-4



HFEP HSTATUS SERVICE

FIGURE III-5

3.0 Network Interface

The X.25 network interface used by the HFEP is described by three tables. The first of these is entitled "HFEP Network Input Events". This is a listing of all possible network input events from an X.25 network that could affect the state of the HFEP protocol machine. This table relates the generalized network event name as used in the formal specification to the corresponding event name within an X.25 network. The table includes commentary about each event.

The second table is entitled "HFEP Network Output Events." It includes all of the X.25 network events that the HFEP signals to the network. The event name as used in the formal specification is listed along with the corresponding event name as specified by X.25.

The third table, entitled "X.25 Services Required by HFEP", lists suggested X.25 operating parameters and options within all possible X.25 services allowed by the CCITT X.25 Recommendation [CCITT80]. These parameters and options are suggested to allow the HFEP to operate properly and reliably over a wide variety of X.25 equipment and environments. However, since the HFEP environment is local to a given node on a network, certain of the parameters could be modified by the implementor. For example, a maximum packet size of 128 octets is specified. To provide better HFEP thruput, this could be made considerably larger (e.g., 1024) if both ends of the HFEP link are capable of generating and receiving such large packets and sufficient buffering is provided. However, an HFEP implementation with such large buffers could not be used to support a remote front end over a public X.25 network since not all public X.25 networks support packet sizes of more than 128 octets.

Finally, it should be noted that with little or no modification the HFEP could be used with network technologies other than X.25. Any other network technology must provide a reliable, multiplexed, individually flow controlled communications path with provision for "expedited data service". The only part of the HFEP specification that requires a service unique to X.25 is the use of the M (More data) bit. The M bit was used in the HFEP design to achieve high channel efficiency in the face of small (128 octet) packets. Its use allows for a multipacket HDATA pdu with Protocol Control Information (PCI) (also known as a packet "header") overhead only on the first packet. For implementations using network technologies allowing larger packets, multipacket pdu's could be eliminated. The new HFEP protocol would then be a proper subset of this current one.

HFEP NETWORK INPUT EVENTS

HFEP Event Name	X.25 Name (DTE side)	Comments
NCON.indication	(Incoming Call)	- Used for creating HFEP machine (Not formally part of HFEP)
NCON.confirmation	(Call Connected)	
NDIS.indication	(Clear Indication)	- Other side requested disconnect
NDIS.confirmation	(DCE Clear Conf.)	- Confirmation of requested disconnection (Not formally part of HFEP) (Ignored by HFEP)
NDATA.indication	(DCE Data)	- Used for conveying most PDU's
NINTRUPT.indication	(DCE Interrupt)	- Used for HCRI PDU
NINTRUPT.confirmation	(DCE Interrupt Conf.)	- (Not formally part of HFEP) (Ignored by HFEP)
NRST.indication	(Reset Ind.)	- Indicates error
NRSTRT.indication	(Restart Ind.)	- Indicates error

HFEP NETWORK OUTPUT EVENTS

HFEP Event Name	X.25 Name (DTE side)	Comments
NCON.request	(Call Request)	
NCON.response	(Call Accepted)	- Sent as part of initializing HFEP machine
NDIS.request	(Clear Request)	
NDIS.response	(DTE Clear Conf.)	- Not formally part of HFEP
NDATA.request	(DTE Data)	- Used to convey most HFEP PDU's
NINTRUPT.request	(DTE Interrupt)	- Used to convey HCRI PDU
NINTRUPT.response	(DTE Interrupt Conf.)	- Not formally part of HFEP

X.25 SERVICES REQUIRED BY HFEP

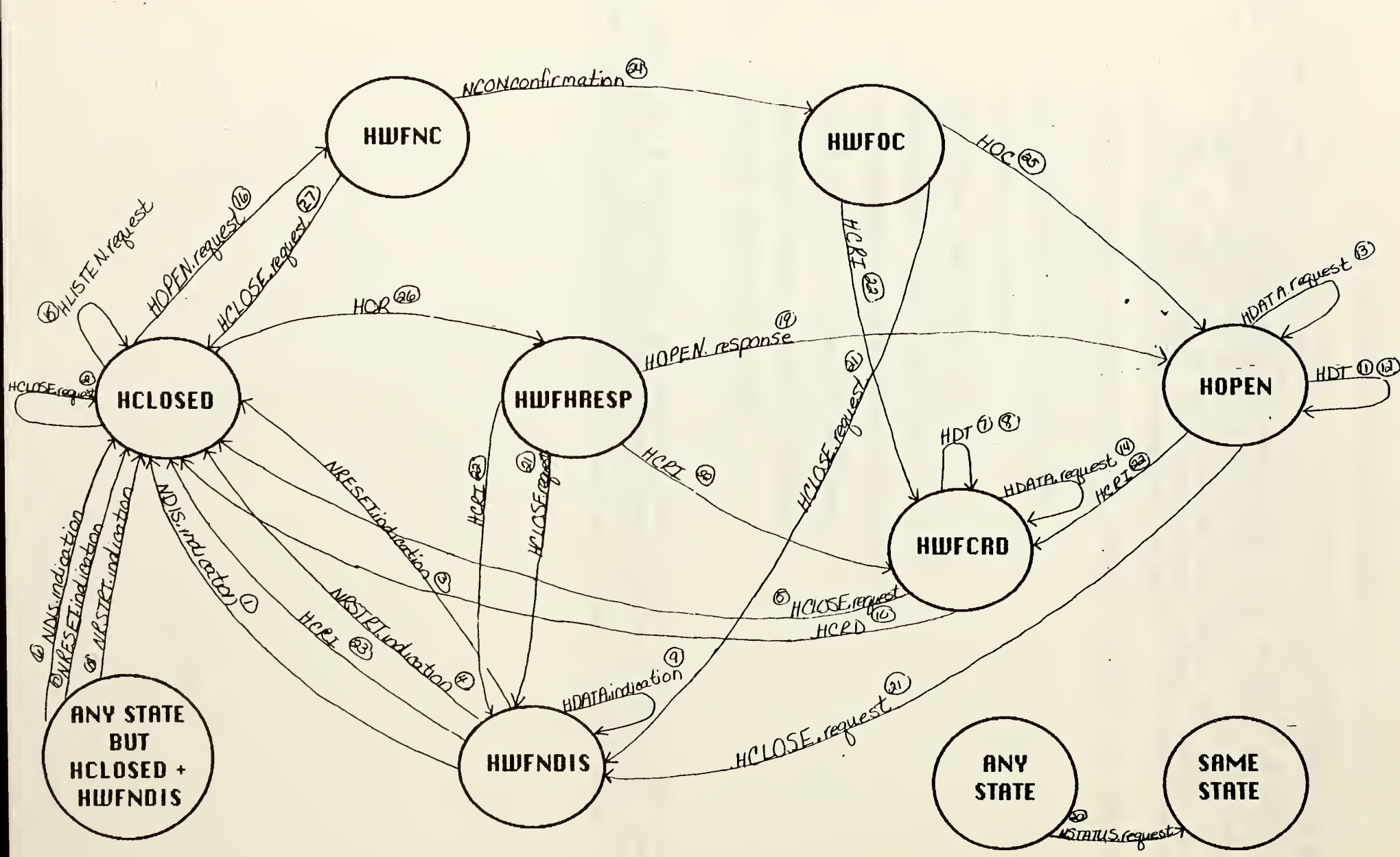
The HFEP uses a minimal subset of X.25 features (to provide the maximum compatibility with existing and future equipment) and complies with Federal Information Processing Standard 100 (FIPS-100) [FIPS83]. The following constraints (some of which are also provided by FIPS-100) are placed on the X.25 services used in HFEP.

- LAPB Protocol with HDLC Framing is used.
- Virtual Circuit Service is used. Neither datagram nor permanent virtual circuit services are used.
- The D (Delivery) Bit is not used.
- The Q Bit not used.
- The Frame Window Size is 7.
- The Packet Window Size is 2.
- The Maximum Packet Size is 128 octets.
- The Host Side is the DTE (Address 03).
- The Front End is the DCE (Address 01).
- No X.25 "Facilities" Used

Note: The X.25 features and services used by HFEP are available on all X.25 networks. Thus, while performance and reliability might be somewhat diminished using HFEP over a public X.25 network, it is possible to do so. The network address would be inserted in the X.25 CALL packet.

4.0 State Diagram

Figure III-6 contains the Finite State Diagram (Finite State Machine) for the HFEP. The states are represented as circles labeled appropriately. Transitions are indicated by arrows. The stimulus causing the transition is indicated by the label on the arrow. The circled number associated with each arrow label represents the transition number in the formal specification.



HFEP STATE DIAGRAM

FIGURE III-6

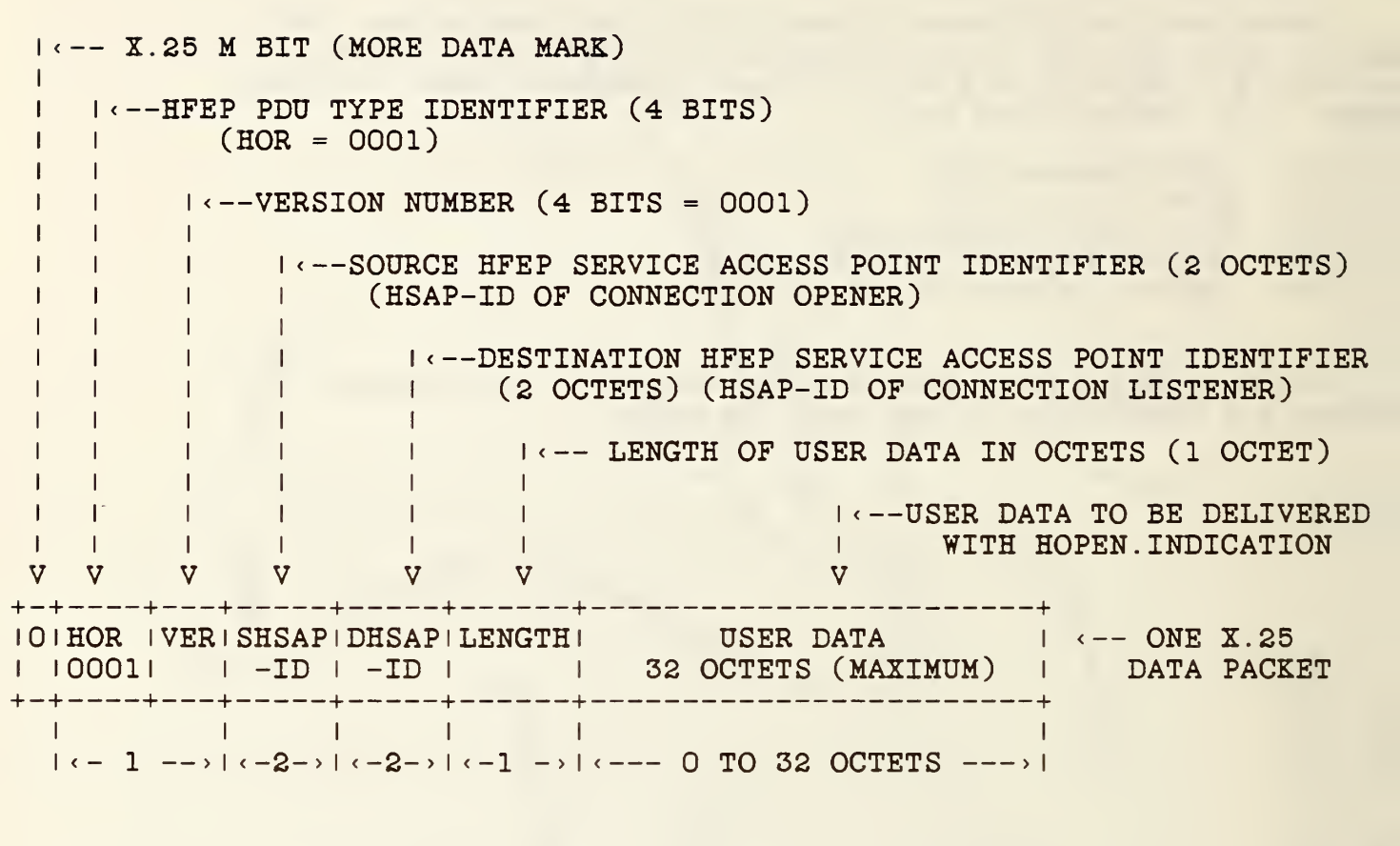
5.0 Appendix A - Protocol Data Unit (PDU) Formats

There are five different types of PDU's defined in the HFEP. Four of these are sent using the normal X.25 data service packets. One (HCRI) is sent in an X.25 interrupt (expedited data service) packet. The packet types are:

HOR - HOPEN Request PDU
HOC - HOPEN Confirm PDU
HDT - HDATA Data Transfer PDU
HCRI - HCLOSE Request PDU (Interrupt)
HCRD - HCLOSE Request PDU (Data)

The formats of these pdu's are described in Figures A-1 through A-4 that follow on the next few pages.

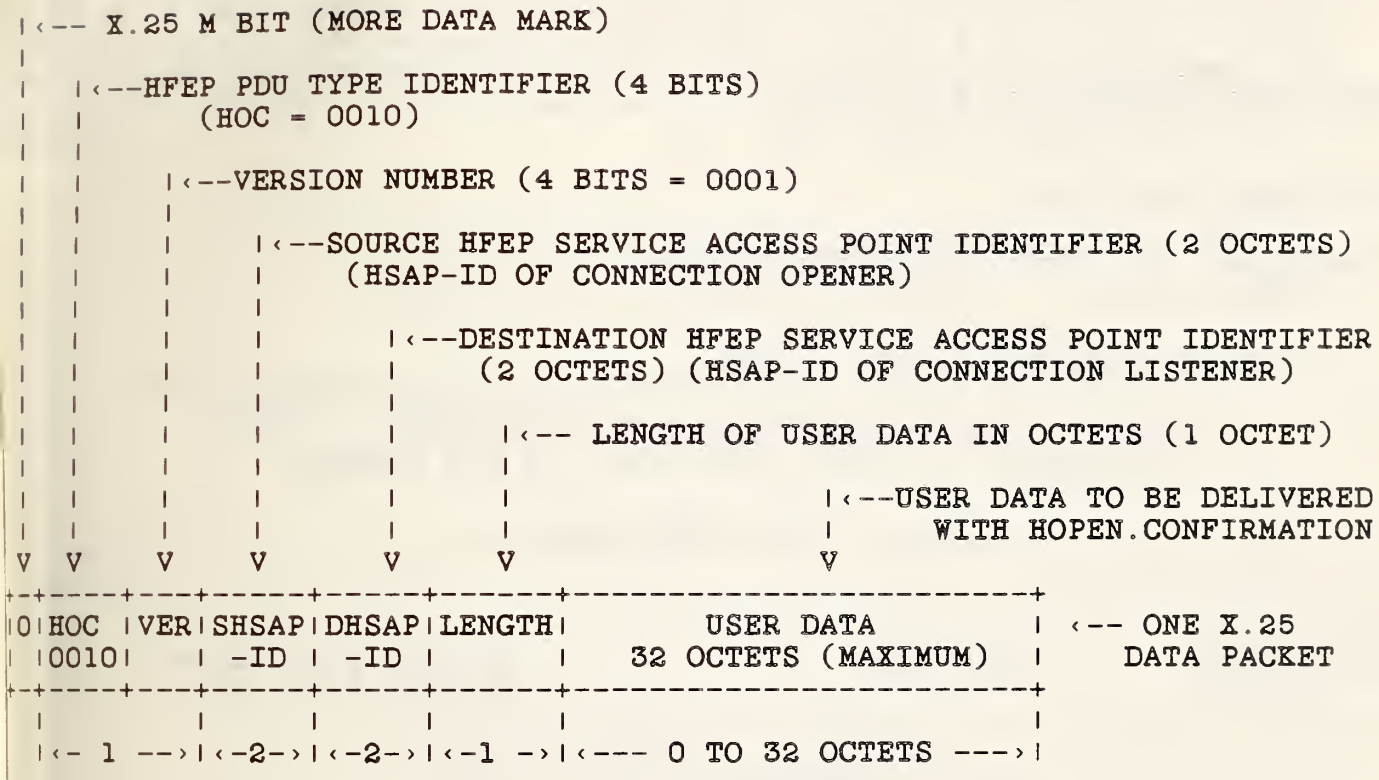
HOPEN REQUEST PDU (HOR)
(SENT IN X.25 DATA PACKET)



HOPEN REQUEST PDU CONSISTS OF ONE X.25 DATA PACKET

Figure A-1

HOPEN CONFIRM PDU (HOC)
(SENT IN X.25 DATA PACKET)



HOPEN CONFIRM PDU CONSISTS OF ONE X.25 DATA PACKET

Figure A-2

HDATA PDU (HDT)

An HFEP DATA PDU (HDT) consists of one or more X.25 data packets. The length field of the first packet header indicates the length in octets of all user data in all the packets that are part of the HDT. The HFEP DATA PDU is one X.25 "message". Therefore, all but the last packet of the HDT are sent with the X.25 M bit (more data mark) on.

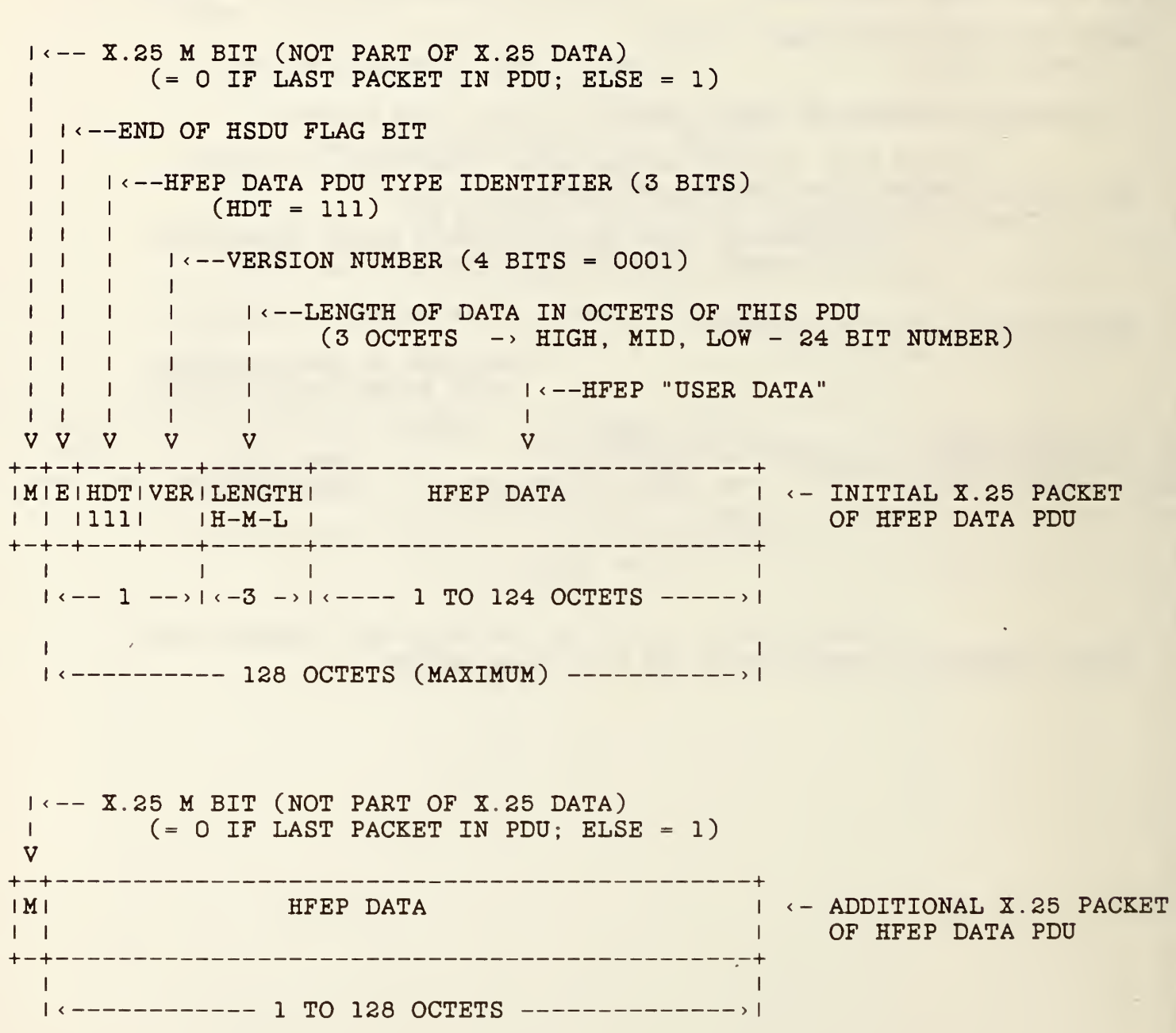
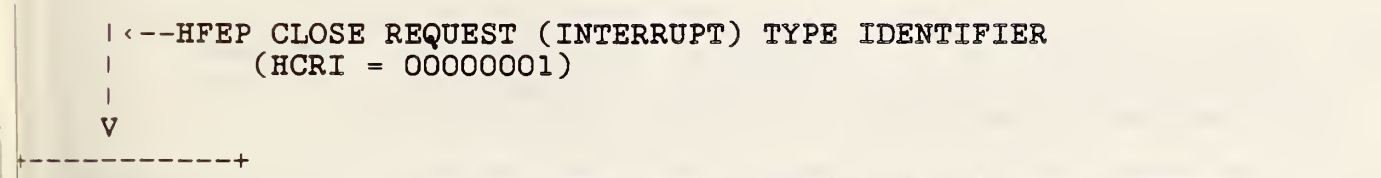


Figure A-3

HCLOSE SERVICE PDUS

The HCLOSE Service requires two PDUs. The first to be sent is the HCRI PDU, and is sent, NOT in an X.25 data packet but rather in an X.25 interrupt packet. The interrupt packet is used to avoid normal flow control on an X.25 connection. The second packet to be sent is the HCRD PDU, and is sent in an X.25 data packet. The second packet is required because the HCLOSE service requires the delivery of user data and the X.25 interrupt packet consists of only one octet. (The above is true for CCITT Recommendation X.25-1980. However, future versions of X.25 may allow for longer interrupt packets.)

HCLOSE REQUEST PDU (HCRI)
(SENT IN X.25 INTERRUPT PACKET)



HCLOSE REQUEST PDU (HCRD)
(SENT IN X.25 DATA PACKET)

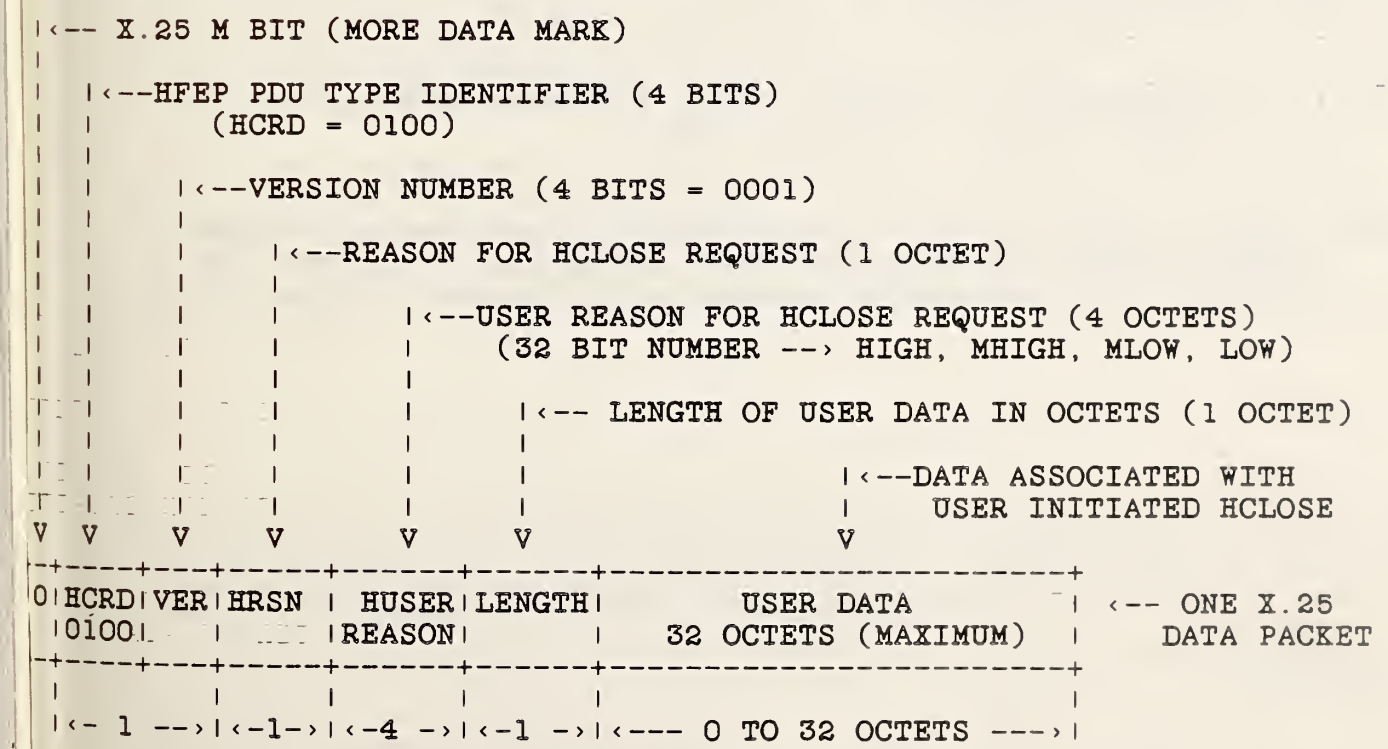


Figure A-4

References

[CCITT80] "CCITT Recommendation X.25 (1980); Interface Between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) for Terminals Operating in the Packet Mode on Public Data Networks", International Telegraph and Telephone Consultative Committee (CCITT) of the International Telecommunications Union (ITU), 1980.

[FIPS83] "Interface Between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) for Operation with Packet-Switched Data Communications Networks", Federal Information Processing Standards Publication 100 (FIPS Pub 100), Federal Standard 1041, July 1983.

[ICST83] Specification of a Transport Protocol for Computer Communications, National Bureau of Standards, Institute for Computer Science and Technology, January 1983.

[IS7498] "Information Processing Systems - Open Systems Interconnection - Basic Reference Model", International Standard 7498, International Organization for Standards.

[IS8073] "Information Processing Systems - Open Systems Interconnection - Transport Protocol Specification", International Standard 8073, International Organization for Standardization.

[Padil84] Introduction to Proposed DOD Standard H-FP, M. A. Padilipsky, RFC 928, December 1984.

[Tenn81] Tenney, Richard, and Thomas Blumer, "An Automated Formal Specification Technique for Protocols," Report No. ICST/HLNP-81-15, National Bureau of Standards, 1981.

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 85-3236	2. Performing Organ. Report No.	3. Publication Date August 1985

4. TITLE AND SUBTITLE

An NBS Host to Front End Protocol

5. AUTHOR(S)

C. Michael Chernick

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

"Front end" processors can be used to "offload" communications processing from host computers, but protocols are needed to communicate between the host and the front end processors themselves. This paper describes a generic protocol (denoted HFEP) for host to (and from) front end communications processors. The HFEP, used in conjunction with additional, more user oriented protocols, such as ISO Transport or Virtual Terminal Protocol, can be used to offload these protocols.

The NBS HFEP provides for a reliable, multiplexed, connection oriented services with a mechanism for process rendezvous. Primitives are defined for opening and closing connections, transfering data and determining the status of a connection. The HFEP uses underlying X.25 network technology (although other reliable, multiplexed and individually flow controlled network connection oriented technologies could be used.)

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

communications processor; communications protocols; computer networks; front end processor; OSI Transport; offload; X.25 technology

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES 92

15. Price $11.50