NIST Technical Note 1905

The Influence of Realism on Congestion in Network Simulations

Christopher Dabrowski
Kevin Mills

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.TN.1905

National Institute of Standards and Technology
U.S. Department of Commerce

NIST Technical Note 1905

The Influence of Realism on Congestion in Network Simulations

Christopher Dabrowski
Kevin Mills
Advanced Network Technologies Division
Information Technology Laboratory

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.TN.1905

January 2016

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie May, Under Secretary of Commerce for Standards and Technology and Director

National Institute of Standards and Technology Technical Note 1905
Natl. Inst. Stand. Technol. Tech. Note 1901, 62 pages (January 2016)
http://dx.doi.org/10.6028/NIST.TN.1905
CODEN: NTNOEF

The Influence of Realism on Congestion in Network Simulations

C. Dabrowski and K. Mills
{cdabrowski, kmills}@nist.gov

Abstract. Many researchers have used simulation to investigate the spread of congestion in networks. These researchers often find that congestion can be modeled as a percolation process, spreading slowly under increasing load until a critical point. After the critical point, congestion spreads quickly throughout the entire network. The researchers also identify various measureable signals that arise around the critical point. These findings appear quite promising as a theoretical basis for monitoring regimes that network operators could deploy to warn of impending congestion collapse. Yet questions surround the extant research because the findings arise from models that are quite abstract. Such models bear little resemblance to networks deployed based on modern technology.

We explore these questions by examining the influence of realism on the spread of congestion in network simulations. We begin with an abstract network simulation, taken from the literature, and add elements of realism in various combinations, culminating with a high-fidelity simulation, also taken from the literature. By comparing patterns of congestion among combinations, we make four main contributions. First, we illustrate that congestion spread in abstract network models differs significantly from spread in realistic models. Second, we show that models investigating network congestion must include specific elements of realism before acceptable engineering findings can be established. Third, we identify the influence of specific elements of realism on congestion in network simulations. Finally, we demonstrate an effective means to compare congestion patterns among network simulations comprising diverse configurations. We hope our contributions lead to better understanding of the influence of realism on congestion in network simulations, and to improved dialog throughout the diverse community of researchers who rely on network simulations.

Keywords: congestion, criticality, networks, percolation, simulation

Table of Contents

List of Tables

List of Figures

1. Introduction

In the past decade or so, the science of complex networks has matured to the point where one can rigorously study the mathematical structure of many classes of probabilistic graphs (e.g., from random to scale-free), as well as dynamic processes moving within such graphs. Network science, which is quite general, has been used to study food webs, social networks, information networks, electric grids, communication networks, traffic networks, metabolic networks, protein and genetic networks, epidemic spreading, and even brain networks. The diversity of networks studied attests to the generality and power of the mathematics that underlie network science.

Typically, significant levels of abstraction are adopted in order to model real networks using techniques (including graph theory [1] and percolation theory [2]) available from network science. Sometimes, researchers disagree on the applicability of such abstractions to the real networks under study. For example, after positing that the Internet has a scale-free topology, Albert, Jeong and Barabasi [3] used network science to demonstrate that the Internet is vulnerable to attacks that strategically remove highly connected nodes, which might represent large interconnection points in selected geographical areas. They also demonstrated that a scale-free topology, such as the Internet, is resilient to random failures. Based on these studies, Albert et al. described the Internet as "robust yet fragile". In a later study, Doyle et al. [4] took exception to the findings of the Albert team, pointing out that many classes of topology could be represented as scale-free graphs, but that not all such classes could be deployed on the Internet, due to specific technological constraints. In particular, router-based communication networks are designed to serve users who inhabit the network edge, where high-degree vertices will thus be found. Further, communication networks are designed so that traffic entering at network edges can be carried efficiently across the backbone to corresponding edges, yielding high throughput. Doyle et al. labeled such topologies as HOTnets: highly optimized networks with organized tolerance and tradeoffs. Further, they argued that HOTnet topologies are robust to attacks against high-degree nodes; as such attacks would remove edge nodes. On the other hand, HOTnets are fragile to failures at low-degree core nodes, which are required to carry transit traffic between network edges. The main point raised by Doyle et al. is that probabilistic graph models of networks might prove too abstract to represent constraints present in the topologies of real communication networks.

The debate between the Albert and Doyle teams shows a tension that exists when the powerful abstractions of network science are applied to study real networks. How can one be sure that chosen abstractions adequately embody key properties of a network under study? This general question motivates the work reported in this Technical Note. Here, though, our focus is on the spread of congestion within a network topology, rather than on vulnerability to failures and attack.

Many researchers [5-14] have used simulation to investigate congestion spread in network topologies. These researchers often find congestion can be modeled as a percolation process on a graph, spreading slowly under increasing load until a critical point. After the critical point congestion spreads quickly throughout the entire network. The researchers also identified various measureable signals that arise around the critical point. These developments appear quite promising as a theoretical basis for monitoring

regimes that network operators could deploy to warn of impending congestion collapse. Despite showing promise, questions surround the extant research. The network models are quite abstract, bearing little resemblance to communication networks deployed based on modern technology.

To explore the adequacy of abstract models, we examine the influence of realism on congestion spread in network simulations. We do not focus on the adequacy of the topological model, as the Doyle team did. Instead, we use the same fundamental graph for all of our simulations. Our chosen graph was adapted from the network topology of an unnamed Internet service provider. Using this HOTnet topology, we focus on subsidiary technological traits: relative speed of nodes, propagation delays, bounds on packet queues, distribution of traffic sources and sinks, correlation among packet injections, and regulation of transmission rates. These traits are fundamental to real, deployed communication networks; yet studies based on abstract models often adopt unrealistic assumptions about such traits.

We begin with an abstract network model, taken from the literature [14], and add elements of realism in various combinations, culminating with a high-fidelity model, also taken from the literature [15]. By comparing patterns of congestion among combinations, we explore a number of questions: Does spreading congestion in abstract network models mirror spreading congestion in realistic models? How do particular elements of realism influence congestion spread? Are any elements of realism essential to capture in models of network congestion? Are some elements of realism unnecessary? What measures of congestion can be insightful across a diverse set of network models?

We make four main contributions. First, we illustrate that congestion spread in abstract network models differs significantly from congestion spread in realistic models. Second, we show that models investigating network congestion must include specific elements of realism before acceptable engineering findings can be established. Third, we identify the influence of specific elements of realism on congestion spread. Finally, we demonstrate an effective means to compare congestion patterns among network simulations comprising diverse configurations. We hope our contributions lead to better understanding of the influence of realism on congestion in network simulations, and to improved dialog throughout the diverse community of researchers who rely on network simulations.

The remainder of this Technical Note is organized into six main sections.

- Section 2 reviews related work where researchers use abstract models to investigate congestion spread in network simulations. Our review shows the scope of extant research, and highlights some of the promising findings regarding signals that arise near a critical point.

- Section 3 describes three simulation models relevant to our experiment. One model, EGM [14], named from initials of the authors (**E**chenique, **G**omez-Gardenes and **M**oreno), is the baseline for the most abstract simulation we use. A second model, MesoNet [15], is the baseline for the most realistic simulation that we use. Previously, MesoNet was used [16] to compare eight congestion-control algorithms designed to replace the standard transmission-control protocol (TCP) deployed throughout today's Internet. A third model, which we designate as

FxNS (or Flexible Network Simulator), extends the EGM model so that elements of realism, extracted from MesoNet, can be included selectively, i.e., turned on/off in various combinations. We define a set of configurations that are valid combinations of FxNS realism elements. We use FxNS for all simulations in our experiment; thus the EGM and MesoNet models define the extremal configurations of FxNS. When no elements of realism are selected, FxNS behaves like EGM. When all elements of realism are selected, FxNS behaves like MesoNet.

- Section 4 details our experiment design. We identify input parameters associated with each realism element extracted from MesoNet. We provide values for those parameters when the corresponding realism elements are enabled and disabled. For input parameters that are always enabled, we assign fixed values. We also identify and define the two parameters that vary during our experiment. Next, we define the four responses that we measure. Finally, in Sec. 4, we describe two forms of self-adaptation, congestion and time-step, which we implemented into FxNS in order to limit consumption of computation cycles and memory.

- Section 5 displays our results as graphs of each response for each configuration at each packet-injection rate.

- Section 6 gives our analysis of the results, and discusses our findings.

- Section 7 gives our conclusions and directions for future work.

2. Related Work

Graph theory [1] provides a framework for rigorous mathematical understanding of complex networks, which can be represented as a set of nodes connected through a collection of links. Graph theory provides a foundation for generating graphs through various random processes, which can lead to differing structures, such as random, small-world, and scale-free networks. Graph theory also provides measures that characterize the structure of networks. Such measures include: (1) node degree, degree distributions and correlations; (2) centrality, diameter and betweeness; (3) clustering, motifs and community structures; and (4) graph spectra. Researchers have applied graph theory to generate artificial networks and to study their structural properties. Researchers have also characterized both manmade and natural networks, using measures provided by graph theory. Graph theory has been extended to include concepts from percolation theory.

Percolation theory [2] enables the study of dynamic processes in spatial frames, such as finite-element grids, which can be represented through graphs. Originally, graph theorists applied percolation theory to study the dynamic properties of graph formation when generated via random processes. The main finding was that there exists a critical point in the probability of creating links in a graph. Below the critical point a graph remains fragmented into self-connected subcomponents. Above the critical point a graph quickly becomes highly connected, forming what is known as a giant connected component (GCC), ultimately spanning all nodes. More recently, researchers have applied percolation theory to investigate dynamic processes that spread among nodes within a preexisting network. Such dynamic processes include cascading failures in electrical grids, evolution of disease epidemics, and expansion of forest fires. Of particular interest for this Technical Note is the use of percolation theory to study congestion spread in communication networks.

2.1 Similarities and Variations among Previous Studies

Studies [5-14] investigating congestion spread in communication networks paint a similar picture. Every study found spreading congestion to be associated with a critical load. Prior to the critical load, congestion was relatively benign. After the critical load, congestion spread quickly throughout the entire network. Further, the studies found a critical load to exist in a wide range of randomly generated topologies (and one real topology), routing schemes, and distribution of packet injectors. In addition, the studies identified various measures that could be used to reflect congestion state in a network. Several studies found measures that changed rapidly as congestion approaches critical load. This suggests that: (1) congestion spread is a percolation process, (2) networks designed to achieve high throughput operate relatively near critical load, (3) congestion collapse occurs quickly in networks that exceed critical load, and (4) measurable signals appear as congestion approaches critical load. If these points are true, then it should prove possible to build monitoring regimes to signal operators when a network nears critical load.

Though each study showed similar findings, the abstract models had variations along four dimensions: topology, traffic sources/sinks, routers and congestion measures. Researchers used either deterministic or probabilistic topologies. The most popular deterministic topology was a square lattice, either open [8, 13] or folded into a toroid [5-

7, 9, 11]. Rykalova et al. [12] also used a ring. Echenique et al. [14] used a real topology taken from the Internet autonomous system map, circa 2001. Arrowsmith et al. [7] started with a square lattice and then generated triangular and hexagonal depleted lattices by probabilistically removing links. Other researchers used random processes to generate topologies: Erdős–Rényi [11], exponential [10], scale-free [10-11], or small-world [11].

Within a topology, researchers used either deterministic or probabilistic processes to place sources, sinks and routers. The most popular approach was to allow every node to be a packet source and sink, as well as a router [9-12, 14]. Sarkar et al. [13] restricted sources and sinks to the network edge, while Mukherjee and Manna [8] placed sources at the top edge of a lattice and sinks at the bottom edge. Other researchers [5-7] assigned nodes to be a source/sink or router with a biased coin flip. All surveyed studies generated loads by having sources inject individual packets, where each packet is destined for a randomly selected sink. The most popular strategy [5-9, 12-13] was for each source to generate a packet per time step (p/ts) with a specified probability. A few studies [10-11, 14] generated a fixed number of p/ts and randomly assigned the packets to sources. One study [10] had a constant density option to ensure a fixed number of packets remained in transit.

In all models surveyed, router nodes queue packets arriving from sources and then forward them at an assigned rate to the next hop along some path toward the sink. Differences appeared with respect to queue discipline, next-hop selection and forwarding rate. The most popular [5-9, 11-12, 14] queue discipline was unbounded first-in, first-out (FIFO) queues. One study [10] used bounded last-in, first-out (LIFO) queues. One study [13] used bounded FIFO queues, where the oldest packet was dropped when a packet arrived at a full queue. Most studies [5-8, 12, 13] selected next hop based on shortest-path first (SPF) in hops. Ties were broken either by shortest queue length [5-6, 13], link usage [7] or tossing a fair coin [8, 12]. One study [9] selected next hop with the choice among three different SPF metrics: hops, queue length, or their sum. Two studies [11, 13] used SPF based on a weighted sum of hops and queue length. One study [10] used guided random walk to select next hops. In most studies [5-7, 10, 13-14] each router forwards one p/ts. In two studies [9, 12] each router forwards one p/ts for each queue. One study [8] has each router forward a batch of packets at each time step. One study [11] assigns routers variable forwarding rates using any of three options: (1) node degree, (2) node betweeness or (3) node betweeness divided by number of nodes in the topology.

The surveyed research used various measures of network congestion, and often multiple measures per study. Congestion measures included: one-way packet latency [5-6, 8, 10]; packets delivered (i.e., aggregate throughput) [5-7]; queue lengths [6-8, 10]; packets in the network [9, 11-12, 14]; and packet drop rate [13]. Various studies analyzed the measures as time series, proportions, or variances.

2.2 Detailed Summaries of Previous Studies

Sole and Valverde [5] studied congestion spread in a square lattice, where each node has four nearest neighbors, and periodic boundary conditions close the lattice into a toroid. Nodes are designated randomly as one of two types: host (probability p=0.08) or router (probability 1-p). Hosts can generate and consume packets, while both hosts and routers can store-and-forward packets. Each node contains a queue of unbounded length. At each time step a host creates a packet with probability λ. Another host is selected

randomly (uniform probability) as the packet's destination, and the packet is appended to the end of the forwarding queue within the creating host. At each time step each host and router also removes a packet (if present) from the front of its queue and then selects which outgoing link is best, relative to the packet's destination, and forwards the packet. The selection process considers both shortest path and congestion: select the neighbor nearest the destination, but in case of ties prefer the neighbor with shortest queue. Sole and Valverde experiment with lattices of two sizes: 32x32 and 256x256. They plot one-way packet delay and number of packets delivered (i.e., throughput) within three different measurement intervals. In the smaller lattice, they find a critical load ($\lambda_c = 0.2$). They show that as load passes λ_c packet delay increases quickly and the number of delivered packets falls gradually. In the larger lattice they find similar behavior, but the critical load shifts ($\lambda_c = 0.055$). They conclude that information transfer is maximal at the critical load, but unpredictability in delays is also maximal, as measured by increased variance. They argue that their model captures *some* essential properties of the Internet, and go on to suggest that the Internet might self-organize into a critical state, where both efficiency and unpredictability are maximal.

Woolf et al. [6] started from the study of Sole and Valverde, but introduced an option for long-range dependence (LRD) in packet generation. LRD ensures that packet arrivals are correlated for periods of time. Absent LRD, packet arrivals are independent. Woolf et al. argue that LRD more accurately reflects user behavior on the Internet. They experiment with a 32x32 lattice ($p = 0.16$) and, like Sole and Valverde, find a critical load ($\lambda_c = 0.39$) after which queue lengths and packet delays increase dramatically and throughput collapses. The main effect of LRD is to increase packet delays prior to the critical load. This stands to reason, since LRD increases variance in the packet-injection rate. Woolf et al. suggest that limiting queue lengths could control congestion. They experiment with bounds of 10, 100 and 1000 packets. Here, bounding queue lengths means that a host may not inject packets into a full queue, which serves to cap the maximum load that can be placed on the network. In such cases, they find that packet delays still increase as load approaches λ_c, but delays and throughput level off at the maximum load.

Arrowsmith et al. [7] conducted follow-on to the Woolf study. Here, in addition to a square lattice, topologies include depleted lattices of two types: hexagonal and triangular. Depleted lattices are created by probabilistically removing links from a square lattice. The researchers claim this is an intermediate step toward studying other irregular (randomly generated) networks, such as scale-free topologies. When forwarding packets, the next router is chosen based on shortest path; ties are broken with a random choice. All queues are unbounded. Arrowsmith et al. find many of the same behaviors reported in the Woolf study. They also show that critical load shifts with topology, occurring earlier in the hexagonal grid than in the sparser triangular grid. In general, Arrowsmith et al. conclude that critical load increases with the sparseness of topology. They also report that measures of queue length are indicative of congestion state in all three topologies, while measures of throughput are not as indicative.

Mukherjee and Manna [8] studied congestion in square lattices that are not folded into a toroid. They study lattices that are 8x8, 64x64 and 128x128. Here, packets are injected into nodes at the top edge of the lattice and then flow downward through the lattice toward nodes at the bottom edge, where they are consumed. In this study, packets

are injected with a probability (ρ) at each top-edge node at each time step. Packets are queued (unbounded length) in nodes, and at each time step a maximum of m packets are forwarded FIFO in bulk to one of two (randomly chosen) neighbors. This setup creates a system where $\lambda_c = \rho = m$. They measure load, defined as the aggregate number of packets queued in the system, at each time step. The system transitions to a congested state when $\rho > m$. Mukherjee and Manna show the distribution of packet-delays to be lognormal (heavy-tailed) and the power spectrum of the queue-length time series to exhibit 1/f-like noise for measurement intervals spanning three orders of magnitude, which they assert is similar to Internet traffic. As with previous studies, Mukherjee and Manna find that queues and packet delays grow quickly as load passes λ_c, and variance in packet delays also increases.

Lawniczak et al. [9] studied congestion spread in a 16x16 lattice, not folded into a toroid, where each node is both a source/sink and router. Here, each node has two FIFO unbounded queues: incoming and outgoing. Each simulation runs for 8000 time steps. At each time step the model takes five actions: (1) update routing tables, (2) create (probability λ) and forward packets, (3) process incoming queues, (4) evaluate network state, and (5) update simulated time. They measure the number of packets in transit, while comparing the effects of three different SPF routing algorithms: (1) hops, (2) queue lengths, and (3) sum of hops and queue lengths. The Lawniczak study finds that critical load varies with the routing algorithm: $\lambda_c = 0.045$ for SPF hops and $\lambda_c = 0.085$ when queue lengths are considered. This implies that congestion-based routing can handle somewhat more load, but they also found that such routing increases correlation in the number of packets in transit.

Tadic et al. [10] studied congestion in two randomly generated irregular topologies: (1) a correlated cyclic (i.e., highly clustered) scale-free network and (2) an uncorrelated homogeneous (i.e., weakly clustered) exponential network. Both topologies were generated randomly using schemes developed by graph theorists. The generating processes allowed the topologies to consist of self-connected subcomponents, rather than a fully connected graph. To account for this, choice of packet destinations ensured that each packet could flow within a single subcomponent. The routing strategy used is somewhat unorthodox. If a packet's destination is in the nearest neighborhood, then the packet is delivered. If the destination is in the next-nearest neighborhood, then the packet moves in that direction. Otherwise, the packet moves randomly. This amounts to a guided random walk. Packet generation could take either of two forms: constant density or constant rate. For constant density, ρ packets are always in flight, and a new packet is created for each packet that reaches its destination. For constant rate (similar to all other studies we survey), a fixed number r of packets are created each time step. The Tadic study limits queue lengths to 1000 packets, but adopts an unorthodox LIFO queuing discipline, where the last packet to be queued is forwarded first. The study measures one-way packet delays and queue lengths. In the case of constant-rate traffic, Tadic and colleagues (like previous studies) find that there is a critical load, after which congestion grows quickly. They conclude that the scale-free topology has a lower critical load than the exponential topology.

Wang et al. [11] extended investigation into the influence of topology on congestion spread. They used graph-theoretic schemes to randomly generate three different topologies: (1) Erdős–Rényi random, (2) Strogatz small-world, and (3) Holme-

Kim scale-free with tunable clustering. All nodes in each topology can generate, consume and forward packets, which are stored in unbounded FIFO queues. At each time step, r packets are generated with randomly chosen sources and sinks, and each node forwards c_i packets toward their destination. To set c_i the study uses three different approaches: (1) $c_i = k_i$, (2) $c_i = b_i$ and (3) $c_i = b_i/n$, where k_i is the degree of node i, b_i is the betweeness of node i, and n is the number of nodes. If the next node is the destination, then a packet is consumed. Otherwise the next node is chosen according to an equation where parameter α provides a weighting between ($\alpha = 1$) SPF based on hops (so-called congestion-blind routing) and ($0 \leq \alpha < 1$) SPF based on a combination of hops and queue length (so-called congestion-aware routing). Congestion is measured by proportion of packets in the network, and a critical load is identified as r_c. With $\alpha = 0.8$ Wang et al. compare phase-transition behavior between free-flowing and congested states for various combinations of topology and node-forwarding capacity.

Rykalova et al. [12] investigated congestion in two topologies: a bi-directional ring and a toroidal square lattice. Each node is a source/sink and router. For the ring, each node has two queues: one per direction. For the lattice, each node has four queues: one per neighbor. All queue lengths are assumed to be unbounded. At each time step each node generates a packet for each queue with probability λ. The destination of each packet is chosen randomly (uniform probability). Packets are forwarded (at the rate of one per queue per time step) to a next node following a SPF hops scheme, where ties are decided by tossing a fair coin. For each topology, the Rykalova team finds a critical load, and shows that, as critical load approaches, the number of messages in the network transitions to a highly correlated state. This finding holds for ring and lattice.

Sarkar et al. [13] model a communication network using techniques inspired by statistical mechanics, which include critical phenomena and phase transitions. While other studies showed existence of critical points in network load and transition from free-flowing to congested states, the Sarkar team attempts to relate such findings to statistical mechanics, which is used by some physicists to model thermodynamic systems. Their goal is to develop a control regime based on measuring analogs to thermodynamic parameters, e.g., temperature, pressure, and order parameter. They classify phase transitions into: (1) first order, where the order parameter changes discontinuously and (2) second order, where the order parameter varies continuously during a transition, but the derivative at the critical value is discontinuous. They suggest that by identifying an approaching critical point, control decisions can be taken to move a congesting network toward a more stable free-flowing state. To demonstrate their argument, they model a (non-toroidal) square lattice, where sources and sinks are limited to boundary nodes that populate the edges of the grid. All internal nodes act only as routers. All nodes have finite queue lengths. At each time step each boundary node creates a packet with probability λ. Packet destination is chosen randomly (source can be destination). Each node forwards one packet from the head of its FIFO queue at each time step. Next nodes are chosen using SPF in hops, but ties are broken based on shorter queue length. When a packet arrives at a full queue, the oldest packet is dropped. The study uses normalized packet-drop rate as the order parameter. The study identifies a critical load (λ_c), and proposes a control scheme based on centralized measurement. The scheme leads to computation and dissemination of a global packet-transmission probability distribution, which in effect implements a priority-based queue discipline. The study also identifies significant future

work that includes: *validation of the theoretical results in more complex and realistic network scenarios*, investigation of the effects of topology, analysis of convergence and stability properties of the control scheme, and moving toward a distributed measurement and control scheme.

A study [14] by Echenique, Gomez-Gardenes and Moreno (hereafter EGM) applies many of the concepts covered above, but in the context of a real communication network topology, consisting of 11 174 nodes, taken from the Internet autonomous systems (AS) map circa 2001. The AS topology, though real, is somewhat ill-suited for studying congestion spread. Each node in the AS topology actually represents a lower-level topology that is not included. The AS topology, which allows Internet service providers (ISPs) to interconnect, evolves from business and policy relationships, whereas the lower-level topology of each ISP network is designed based on engineering principals [4] to ensure efficient transfer of packets among customers. For this reason, studying congestion spread in an ISP topology appears more suitable. EGM measured the degree-distribution of the AS map, which shows a scale-free topology that exhibits a -2.2 slope on a log-log plot of degree distribution vs. frequency. Each node in the topology is a source/sink and router. At each time step p packets are injected, with the source and destination chosen randomly (uniform). Each node forwards one p/ts from the head of its unbounded FIFO queue. The routing scheme is the same as that used by Wang et al. [11]. If the next node is the destination, the packet is consumed. Otherwise the next node is chosen according to an equation where parameter h provides a weighting between SPF based on hops ($h = 1$) and SPF based on a mixture ($h < 1$) of hops and queue length (congestion-aware routing). The lower h, the more that queue length is considered. Like Wang et al., EGM measures congestion through the proportion (ρ) of injected packets in the network. EGM find that critical loads (p_c) exist, but shift with the routing scheme. When $h = 1$, they find a second-order phase transition, where congestion begins increasing at a lower value of p_c but increases more gradually. For various values of $h < 1$, they find first-order phase transitions. Lower values of h lead to higher values of p_c and to a greater increase in congestion at the phase transition.

2.3 Questions Arising from Previous Studies

Uncertainty arises because previous studies used models that are quite abstract, bearing an unknown relationship to real communication networks. To judge the degree of abstraction, consider MesoNet [15], a realistic model of an ISP network. While concise (requiring only 20 parameters), MesoNet can represent such elements of realism as: network configuration (topology, router speeds, propagation delays and buffer sizes), distribution and operation of sources and receivers, user behavior (think time, patience and transfer sizes), congestion-control protocols, and simulation and measurement controls. After conducting sensitivity analyses of MesoNet, Mills et al. [16] used the simulator to compare various proposed congestion-control algorithms. In October 2008, the Mills team presented results to engineers for an ISP that provided MesoNet with a topology. In May 2009, they presented results to the congestion-control working group within the Internet Research Task Force. In March 2010, they presented further results to engineers attending the Internet Engineering Task Force. Both researchers and practicing

network engineers found MesoNet acceptably realistic for comparing congestion-control algorithms.

When comparing MesoNet to more abstract network models, one finds several differences. First, MesoNet provides higher levels of realism with respect to engineering factors present in deployed communication networks based on modern Internet technology. And the reasons why various features were included or excluded in MesoNet are documented. Second, MesoNet was subjected to sensitivity analyses in order to understand how variation in parameters changed model behavior. Third, MesoNet and related simulation results were presented to network researchers and practicing network engineers in order to obtain feedback. Abstract network models used to study congestion spread were not subjected to the same level of scrutiny as MesoNet. The studies we surveyed give little rationale for including or excluding particular features in the models used. They did not describe sensitivity analyses conducted on the models, or report outreach activities to obtain feedback from network researchers and engineers.

Though many studies have been conducted, no one seems to know what level of realism should be required in simulations that study congestion in communication networks. This uncertainty leads to the questions that motivate the research reported in the remainder of this Technical Note: Do abstract models include too little realism? Do models like MesoNet include unnecessary realism? How do elements of realism influence the spread of congestion in network simulations? Are some elements of realism essential? Can some elements of realism be ignored?

3. Models

We conducted an experiment (see Sec. 4) with a simulation model that we call FxNS (Flexible Network Simulator). FxNS starts from the base of an abstract model (EGM) developed by Echenique, Gomez-Gardenes and Moreno [14]. We added to EGM a set of seven realism elements, which we took from MesoNet [15]. We implemented the seven realism elements as optional within FxNS. Since each of the realism elements can be enabled or disabled, FxNS could support (2^7 =) 128 configurations. However, as we explain in Sec. 3.3, we chose to respect some dependencies among the realism elements. This means FxNS supports only 34 of the 128 configurations that would be possible if realism elements were independent. FxNS can be configured to behave as EGM (the most abstract model), as MesoNet (the most realistic model), and any of 32 valid combinations of realism elements intermediate between EGM and MesoNet.

In Sec. 3.1 we describe the abstract EGM model [14]. EGM used their model to study congestion behavior in a realistic 11 174-node topology, which is a snapshot of the Internet autonomous system (AS) topology circa 2001. We motivate our selection of the EGM model. We recap congestion spreading behaviors found by EGM.

In Sec. 3.2 we describe MesoNet, including its complete set of 20 parameters spread among five categories. For 18 MesoNet parameters we define fixed values that we adopted for use within FxNS during our experiment. For four of those parameters, as we explain below, we selected values that amount to eliminating them from FxNS. The remaining two MesoNet parameters are variable within our experiment, though both are determined by a single EGM parameter: packet-injection rate (p).

In Sec. 3.3 we define our mapping from MesoNet parameters to FxNS realism elements. We also identify dependencies we adopted between FxNS realism elements and we give justifications for adopting those dependencies. Finally, we detail our technique for labeling FxNS configurations, and then list the 34 valid FxNS configurations.

In Appendix A we provide simulation results demonstrating that FxNS correctly implements both the EGM and MesoNet models. Using FxNS with all realism elements disabled, we replicate EGM results [14] for the 11 174-node topology, which we obtained from the original developers of EGM. We then replace this large topology with a smaller (218-node) topology taken from MesoNet, and demonstrate that our FxNS implementation of the EGM model produces the same fundamental behaviors as exhibited for the larger topology. To verify that our FxNS implementation of MesoNet realism elements is correct, we compare simulation results from MesoNet against results from FxNS when all realism elements are enabled and p ranges up to 5000.

3.1 Abstract EGM Model

As reported in the literature [14] the EGM model evaluated the spread of congestion in an 11 174-node topology taken from a 2001 snapshot of the Internet AS topology collected by the Oregon Route Views project. We obtained the topology directly from the creators of EGM. Figure 1 shows a visual representation of the topology, created by Sandy Ressler, a colleague in our laboratory. The colors, which have no semantic meaning, represent an attempt by the layout algorithm to assign nodes to clusters. The Echenique team claims that the topology is scale free, as the probability (P_k)

a node has degree k approximately equals $k^{-\gamma}$, with $\gamma = 2.2$. To verify that claim, we plotted (Fig. 2) the node degree (x axis) against the frequency (y axis) on a log-log scale. The claim seems approximately correct.

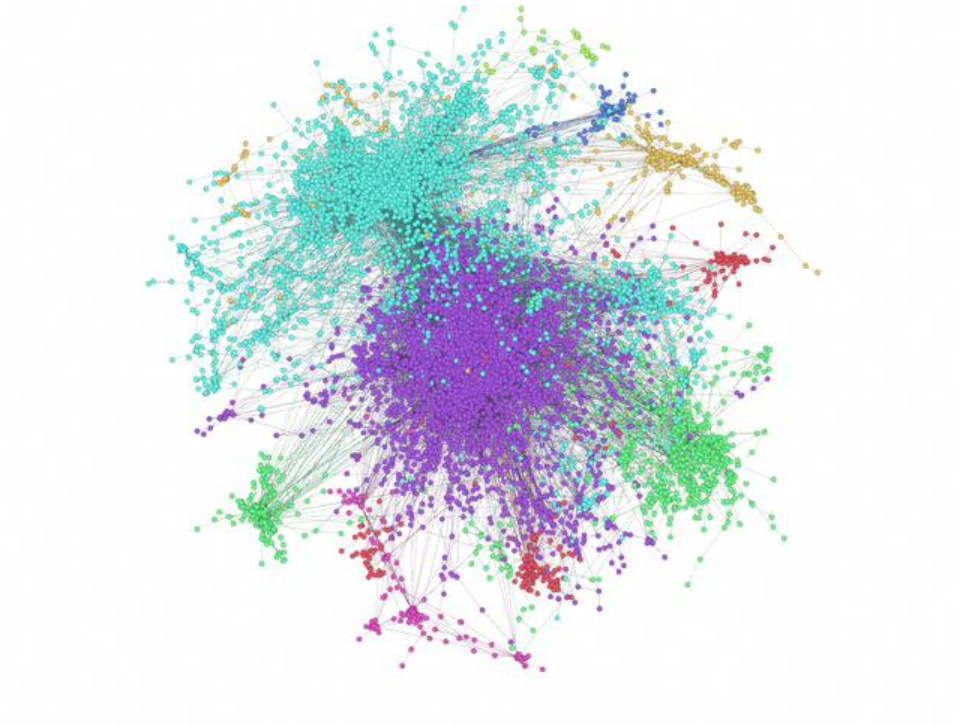

Figure 1. Visualization (courtesy Sandy Ressler) of an 11 174-node Internet AS topology provided by Enchenique et al. [14]

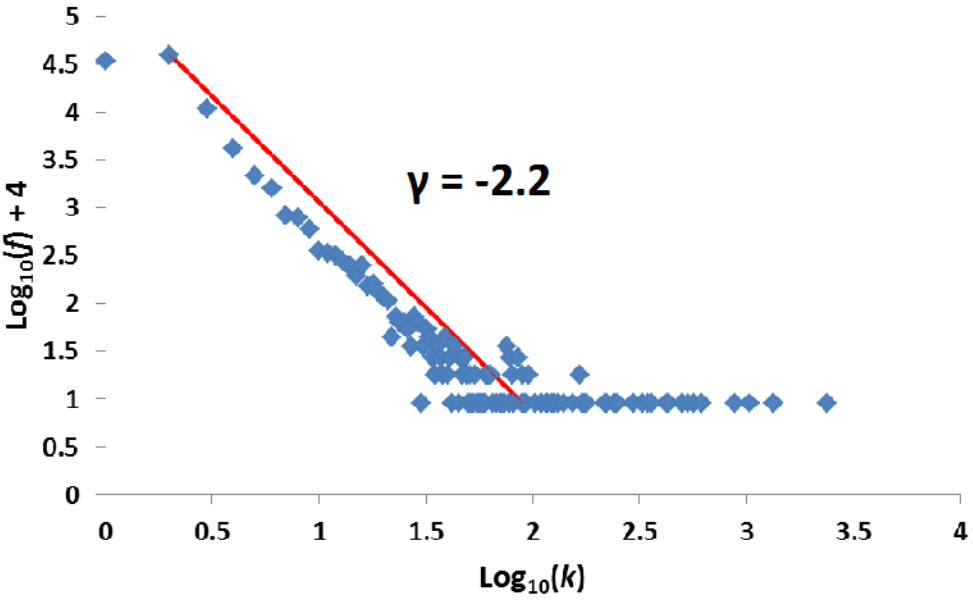

Figure 2. Log-Log plot of node degree (x axis) vs. frequency (y axis) of 11 174-node AS topology

In the EGM model, p packets are injected at each time step with the source node for each packet chosen randomly (uniform), and also with the destination node (which cannot be the source) chosen randomly (uniform). Injected packets are placed at the end of the source node's FIFO packet queue, which can be of infinite length. After injecting packets at a time step, each node can also remove one packet from the front of its queue and forward it to a next-hop node. If a next hop is the packet's destination, then the packet is removed from the system (i.e., delivered); otherwise the next hop is chosen as the neighboring node i with minimum δ_i as defined in eq. 1:

$$\delta_i = hd_i + (1-h)c_i \tag{1}$$

where i is the index of a node's neighbor, d_i is the minimum number of hops to the packet's destination via i, and c_i is the queue length of i. When $h = 1$ the routing amounts to shortest path in hops. When $h < 1$, routing is said to be congestion aware, as packets may follow routes that can be longer in hops, but shorter in total delay incurred due to packet queuing. The lower the value of h the more congestion aware the routing becomes. Should $h = 0$ then routing becomes fully congestion aware.

The EGM model measures congestion as ρ, the ratio of packet outflow to packet inflow. The specific measure is defined in eq. 2:

$$\rho = \lim_{t\to\infty} \frac{A(t+\tau) - A(t)}{\tau p} \tag{2}$$

where A is the aggregate number of packets queued in the network, t is time, τ is the measurement interval size, and p is the packet injection rate.

Echenique et al. [14] used their model to explore effects of various degrees of congestion-aware routing as p increases. In general, they found that, under shortest-path routing by hops ($h = 1$), congestion ρ undergoes a second-order phase transition as p passes a critical load, while under various degrees of congestion-aware routing (h set to 0.95, 0.75 and 0.5) ρ undergoes a first-order phase transition as p passes critical load. Using congestion-aware routing postponed the phase transition: the more congestion-aware was the routing the higher the critical load, at the cost of a bigger step size at the transition. The reason behind this behavior is easy to see: as congestion develops, congestion-aware routing allows alternate routes to be exploited. Once those alternate routes congest, the system has no room to adapt to increasing load, and so congestion increases rapidly.

3.2 Realistic MesoNet Model

MesoNet provides a reasonably realistic model of a TCP/IP (Transmission Control Protocol and Internet Protocol) network, requiring only 20 parameters spread across five categories, as shown in Table 1. One category defines the network configuration, including such engineering details as: topology, propagation delay on backbone links, forwarding speed of network routers and size of buffers in routers. A second category defines the number and distribution of sources and sinks, and the speed with which they can generate and consume packets. A third category defines user traits:

think time, patience, Web-browsing file sizes, and file sizes for larger traffic types. Users may also inject temporary increased load to create spatiotemporal congestion or long-lived flows. The fourth category encompasses parameters that define congestion-control regimes used within the network, including: congestion-control algorithm, initial congestion window, and initial slow-start threshold. The fifth category defines measurement-interval size, simulation duration, and initial startup pattern for sources. Below, we explain the parameters further. Note that Table 1 also shows mapping of MesoNet parameters to FxNS realism factors. We cover that mapping in Sec. 3.3.

Table 1. MesoNet parameters organized in five categories

Category	**ID**	**Name**	**FxNS Realism Factor**
Network	x1	topology	NC (Node Classes)
	x2	propagation delay	DE (Propagation Delay)
	x3	network speed	VS (Variable Speed)
	x4	buffer provisioning	PD (Packet Dropping)
Sources & Sinks	x5	number sources/sinks	SR (Sources & Receivers)
	x6	source distribution	
	x7	sink distribution	
	x8	source/sink speed	VS (Variable Speed)
Users	x9	think time	*p* (Injection Rate)
	x10	patience	n/a
	x11	web object file sizes	FL (Flows)
	x12	larger file sizes	n/a
	x13	localized congestion	
	x14	long-lived flows	
Congestion Control	x15	control algorithm	TCP (Transmission Control Protocol)
	x16	initial *cwnd*	
	x17	*fsst* and *sst*	
Simulation Control	x18	measurement interval	fixed
	x19	simulation duration	fixed
	x20	startup pattern	*p* (Injection Rate)

3.2.1 Network Configuration

A network configuration requires a topology (x1) of routers and links, as shown for example in Fig. 3, adapted from the topology of a modern ISP. MesoNet supports topologies with up to three hierarchical router tiers: backbone routers (A-P in Fig. 3), point of presence (PoP) routers (A1-P2) and access routers (A1a-P2g). To model

heterogeneity in network access, MesoNet allows three types of access routers: D-class (e.g., red nodes in Fig. 3, which connect directly to backbone routers), F-class (e.g., 40 green nodes) and N-class (e.g., 122 small gray nodes). Classifying access routers enables different speeds to be assigned. As discussed later, sources and receivers compose a fourth tier distributed below access routers. Packets flowing between a source-receiver pair follow a single ingress/egress path between an access router and a top-tier backbone router. Propagation delays on backbone links are an intrinsic property of the topology, which also specifies paths taken by packets flowing among backbone routers. Parameter x2 can scale down (e.g., x2 = 0.5) or up (e.g., x2 = 2) propagation delays on all backbone links. In our experiments, when the propagation-delay realism factor is enabled, we use intrinsic delays for backbone links in the topology, and we set x2 to 1.

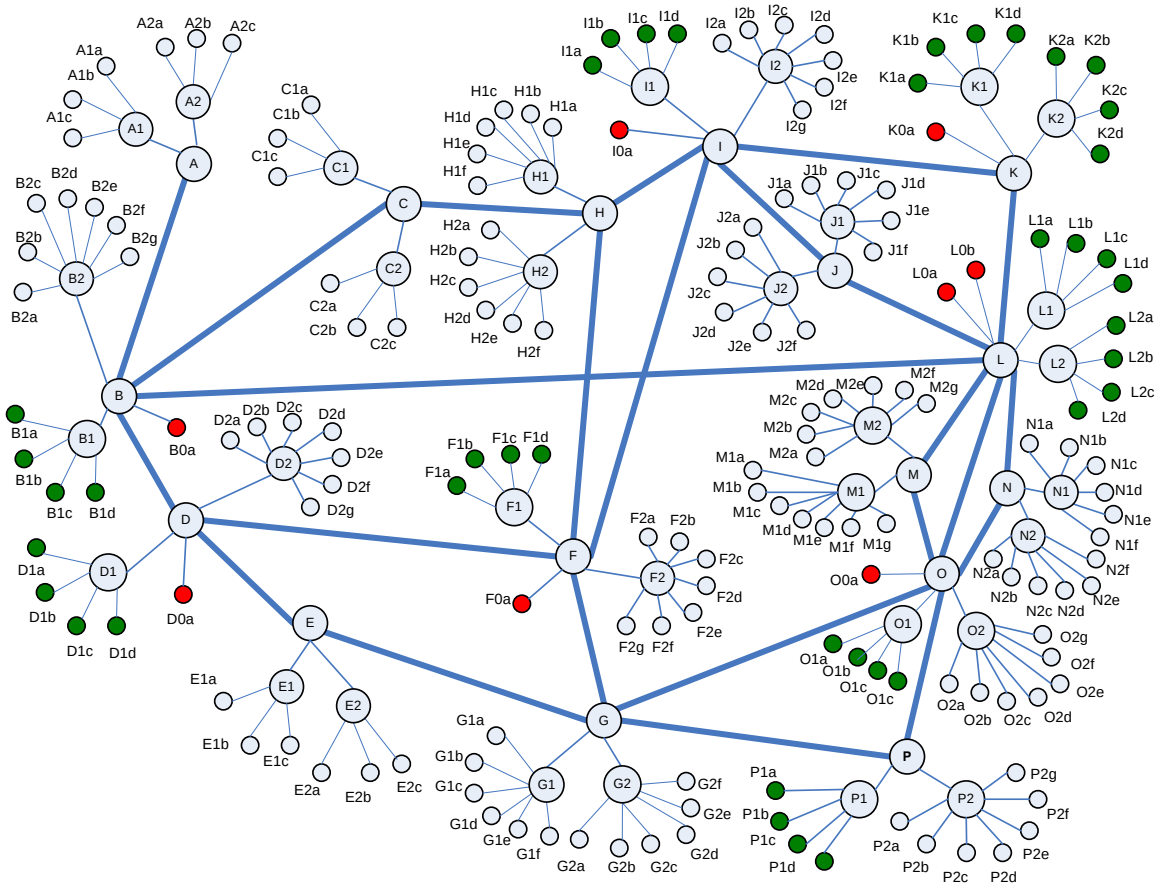

Figure 3. Three-tier topology with 16 backbone routers (A-P), 32 point-of-presence routers (A1-P2) and 170 access routers (A1a-P2g) - 8 red and 40 green access routers may operate at higher speeds than remaining access routers

MesoNet assigns transmission speeds to routers. Each backbone (BB) router multiplexes packet forwarding from a single buffer shared among all attached links, while point-of-presence (PoP) and access (A) routers have two buffers each, one heading toward the backbone and one heading from the backbone. PoP and access routers alternate forwarding between each of the two buffers. Because MesoNet packets have no size, router speeds are assigned in units of packets/time step (p/ts). Six parameters, shown in Table 2 col. 1, define the speeds of all router classes (col. 3), using relationships shown

in col. 4. Note that every defined relationship includes parameter *s*1. By assigning values to the remaining parameters, e.g., as in col. 2, one can establish reasonable engineering relationships among the speeds of the router classes. Then, by equating *s*1 with model parameter x3, speeds of all routers in a topology can be scaled appropriately simply by changing the value of x3, as shown in col. 5, which indicates the speed of each router class in p/ts when x3 = 40 p/ts. We use these values in our experiment whenever the variable-speed realism factor is enabled.

Table 2. Relationships among router classes used to scale speeds

Parameter	**Value**	**Speed Relationships**		**Speed Scaling with X3**
*s*1	X3	**Router Class**	**Speed**	**X3 = 40 p/ts**
*s*2	4	Backbone	*s*1 x *BBspeedup*	80 p/ts
*s*3	10	PoP	*s*1/ *s*2	10 p/ts
BBspeedup	2	**N**-Class	*s*1/ *s*2/ *s*3	1 p/ts
Bfast	2	**F**-Class	*s*1/ *s*2/ *s*3 x *Bfast*	2 p/ts
Bdirect	10	**D**-Class	*s*1/ *s*2/ *s*3 x *Bdirect*	10 p/ts

To provision router buffers, MesoNet allows size (in packets) to be selected using any of four algorithms. In our experiments, when queue lengths are finite, we compute buffer size using one of those algorithms: RTT × capacity. We fix RTT = 250 ts and select capacity by router class from the values shown in col. 5 of Table 2. MesoNet discards packets arriving at a full buffer.

3.2.2 Sources and Receivers

MesoNet requires that a fourth tier of sources and receivers be created and then distributed under access routers. Sources equate to computers that have information of interest to receivers. MesoNet includes a variable, *baseSources*, which is the target number of sources to locate under each access router. MesoNet fixes the number of receivers to be four times *baseSources*. MesoNet parameter x5 serves as a multiplier to scale the number of sources and receivers. For example, given that *baseSources* = 100 and x5 = 3, then 300 sources and 1200 receivers would be attached to each access router – so the topology in Fig. 3, which has 170 access routers, would contain 51 000 sources and 204 000 receivers. These totals are only approximate because MesoNet allows the distribution of sources and receivers to be adjusted, as discussed next.

Recall that access routers come in three classes, as shown in Table 3 col. 1. The precise number of sources under access routers of each type can be adjusted by assigning the probability, *probNs*, a source is under an N-class router and the probability, *probNsf*, a source is under an F-class router. The probability a source is under a D-class router is then 1 – (*probNS* + *probNsf*). For example, if each router class has a target of 300 sources, then the total number of sources below three routers, one of each class, will be (3 × 300 =) 900. Assigning values to *probNs* and *probNsf* would reapportion sources by router class. Similarly, assigning values to *probNr* and *probNrf* would reapportion

receivers. In our experiments, when the sources-and-receivers realism element is enabled, we use the fixed values shown in the caption for Table 3.

Table 3. Sample computation of number and distribution of sources and receivers given the topology shown in Fig. 3 and *baseSources* = 100, x5 = 3, *probNS* = 0.34, *probNsf* = 0.33, *probNr* = 0.34, *probNfr* = 0.33

Class	routers	srcs/router	#srcs	% srcs	rcvrs/router	#rcvrs	% rcvrs	Flows	%
N	122	306	37 332	72.4	1224	149 328	72.4	NN	52.4
								FN	33.3
F	40	297	11 880	23.0	1188	47 520	23.0	FF	5.3
								DN	6.7
D	8	297	2376	4.6	1188	9504	4.6	DF	2.1
								DD	0.2

Each source periodically transfers a flow of packets, after randomly selecting a receiver. The location of a source-receiver pair influences the characteristics of the path for a packet flow. Table 3 col. 9 lists six possible flow classes. Table 3, col. 10 shows the proportion of flows in each class, assuming parameter values shown in the caption.

Sources and receivers can transfer packets to/from the network at some maximum speed. MesoNet includes two settings: *Hbase* and *Hfast*, which specify a number of p/ts. Parameter x8 specifies the probability that a source or receiver connects at a speed of *Hfast*. In our experiment, whenever the variable-speed realism element is enabled, we fix *Hbase* to 0.2, *Hfast* to 2, and x8 to ½.

3.2.3 User Behavior

MesoNet models users as periodically active sources that cycle between thinking and sending. A source selects a random thinking time from an exponential distribution with a mean given by parameter x9. Upon expiration, the source enters a sending state, where a flow of packets is transmitted to a randomly selected receiver. Once all packets are acknowledged, the source selects a new random thinking time. In MesoNet, flows may be associated with end-users who have finite patience or with programs that have infinite patience. Parameter x10 specifies the probability a source has finite patience. In our experiments, think time is replaced by packet-injection rate (*p*) and all users have infinite patience.

When sending, a source selects a Pareto-distributed flow size (in packets) with shape α and mean λ (MesoNet parameter x11). In our experiments, when flows are enabled, we select flow sizes with $\alpha = 1.5$ and $\lambda = 350$ packets. MesoNet also allows sources to transmit larger files. Parameter x12 can be a set to specify those sizes. MesoNet also supports simulation of spatiotemporal congestion (x13) and specific long-lived (x14) file transfers. FxNS does not implement these larger flow-size options.

3.2.4 Congestion Control Protocols

A congestion-control algorithm allows a source to adapt its transmission rate

based on perceived congestion. Parameter x15 specifies probabilities that a specific congestion-control algorithm is assigned to any source. In our experiments, sources implement only the TCP congestion-control algorithm. In outline: TCP [16] probes (during initial slow start) for available transmission capacity by first sending a few packets and then increases the rate quickly as acknowledgments arrive. Upon packet loss, TCP switches to congestion avoidance, reducing transmission rate by 50 % and then increasing the rate slowly on subsequent acknowledgments.

TCP has parameters that control its behavior. Upon connecting, a source first sends a specified number of packets, known as the initial congestion window (x16). As acknowledgments arrive from the receiver, the source increases the *cwnd* exponentially. Absent any losses, a source switches to a logarithmic increase in *cwnd* after reaching a first slow-start threshold (*fsst*). If the *cwnd* increases to a second slow start threshold (*sst*) without loss, then the source switches to congestion avoidance, where *cwnd* increases linearly. The *fsst* and *sst* comprise MesoNet parameter x17. While moving through slow start, a source switches immediately to congestion avoidance upon the first lost packet. In our experiments we set initial *cwnd* to 2, *fsst* to 100, and *sst* to $2^{30}/2$.

3.2.5 Simulation and Measurement Control

MesoNet samples system state at periodic intervals of size *M* ts (x18). Parameter x19 is the number (*MI*) of measurement intervals to be sampled. Simulation duration is $M \times MI$ ts. In our experiments we set $M = 200$ ts and $MI = 1000$, and so each simulation executes for 200 000 ts. MesoNet parameter (x20) defines a startup distribution for sources, allowing load to be present at simulation onset. In our experiments, packet-injection rate (*p*) determines the startup pattern of sources.

3.3 Factored FxNS Model

We factored MesoNet into seven realism elements and then inserted them into FxNS. Section 3.3.1 describes each element. Section 3.3.2 identifies and justifies dependencies among the elements. Section 3.3.3 defines a coding scheme to label FxNS configurations, which are combinations of realism elements, and lists 34 configurations that respect identified dependencies.

3.3.1 FxNS Factors

Table 1 Col. 4 shows how MesoNet parameters map to FxNS realism elements. Each element can be enabled or disabled. Below, for each element, we describe the mapping and the effect of enabling and disabling.

NC: Given a topology, such as Fig. 3, enabling node classification (NC) implies routers are tagged as backbone, PoP, or access. Enabling NC restricts packet injection to occur only at access routers, i.e., the network edge. When NC is disabled routers are homogeneous and packets may be injected at any router.

DE: When propagation delay (DE) is enabled, each backbone link in the core of the topology is assigned a propagation delay consistent with physics and with the

geographic placement of the routers on each end of the link. When DE is disabled, backbone links exhibit no propagation delays.

VS: When variable speed (VS) is enabled, routers are assigned packet-forwarding rates that vary with router class (i.e., backbone, PoP and access) and subclass (i.e., directly-connected, fast or normal) for access routers. These rates are assigned with an engineering relationship that allows higher-level routers to accommodate packets from connected lower-level routers. Here, we assign the rates shown in Table 2. In addition, when VS is enabled and the topology includes source and receiver nodes (see SR below) then those nodes are assigned rates that vary with node type: basic (0.2 p/ts) or fast (2 p/ts). When both VS and SR are enabled, we randomly assign (unbiased coin flip) types to sources and receivers. When VS is disabled, all routers have identical forwarding rate. Here we assign 9 p/ts, which is the weighted average rate of routers in our topology when VS is enabled. When SR is enabled but VS is disabled we assign rates of 9 p/ts to sources and receivers.

PD: When packet dropping (PD) is enabled FIFO buffers are assigned a finite size, computed as 250 ts × router forwarding rate. Packets arriving at a full buffer are discarded. When PD is disabled buffers have infinite capacity, and packets are never discarded.

SR: When sources and receivers (SR) are enabled we include a fourth tier, not shown in Fig. 3, of sources and receivers under access routers. Table 3 gives the number and distribution of sources and receivers. Here, we create 51 588 sources and 206 352 receivers uniformly distributed under each subclass of access router. Enabling SR expands our topology from 218 nodes to 258 158 nodes. Enabling SR restricts packet injection to occur only at sources, and packet removal to occur only at receivers. In fact, enabling SR leads to creation of an independent packet-injection process for each source. When SR is disabled our topology is limited to the 218 nodes shown in Fig. 3. Further, packets are injected from a single injection process within each router.

FL: With flows (FL) enabled, packets are injected as related streams. The packet-injection process is altered to represent arrival of packet streams rather than individual packets. Each source waits for an arrival time, selects and connects to a receiver, selects a flow size, injects packets at whatever rate is appropriate, and then waits for a next arrival time. This cycle continues throughput a given simulation. The size of each packet stream is selected from a Pareto distribution with mean of 350 packets and shape of 1.5. Individual packets in a stream are injected at the rate of the injecting node, but not subject to any congestion-control restrictions unless TCP is enabled (see below). As we explain below, the injection rate p and mean flow size are used to prorate flow arrivals so as to create equivalent packet-injection loads for a given p whether FL is enabled or disabled. Enabling FL also activates a flow-connection process. Before injecting any data packets, the source and receiver in a flow must exchange connection request and accept packets. A retry procedure is implemented, with exponential back off. The FL retry procedure uses the same parameters normally adopted for real Internet TCP flows. If a source sends three connection requests without receiving any connection accept from a receiver within a prescribed time, then the flow is aborted and the source waits for its next arrival time. When FL is disabled packet injection occurs without considering streams, stream sizes, or connection/retry.

TCP: When TCP is enabled the rate of packet flow in each stream is regulated with congestion-control procedures. At stream onset slow-start procedures are activated. The number of packets defined by the initial *cwnd* is injected at whatever rate is possible for the source. As acknowledgments are returned from the receiver, the number of packets that can be sent increases exponentially until *fsst*, after which the increase is logarithmic until *sst*. If *cwnd* reaches *sst*, then congestion-avoidance procedures are activated and *cwnd* increase becomes linear. Upon first packet loss, *cwnd* is cut in half and congestion-avoidance procedures are activated. If no acknowledgments are received within a prescribed time, the *cwnd* is cut in half and the *sst* is set to that value. Subsequently, slow-start procedures are activated. Once each data segment has been acknowledged, the flow is terminated and the source waits for a next arrival time. When TCP is disabled and FL is enabled, after a source and receiver connect, the source injects the stream of packets into the network at the rate assigned to the source (i.e., there is no congestion control and no packet acknowledgment).

Simulation Duration: FxNS implements MesoNet's measurement-interval size (*M*) and simulation duration (*MI*). Here we fix *M* to be 200 ts and we fix *MI* to be 1000. As a result, simulation durations in our experiments are fixed to ($M \times MI$ =) 200 000 ts. But, as explained in Sec. 4.4, when PD is disabled, simulation duration can self-adapt to a smaller value in order to prevent exceeding the memory available on a hosting computer.

Packet-Injection Rate: FxNS subsumes two MesoNet parameters (think time and startup pattern) with packet-injection rate (*p*). MesoNet simulates flows that arrive after a think time expires. When FL is enabled FxNS replaces think time with an arrival process that computes the probability P(*n*, *t*) that a flow arrives at injection source *n* at time step *t*. Specifically, $P(n, t) = p/(N \times f)$, where *N* is the number of potential injection sources and *f* is the average flow size in packets. Rather than implement a separate startup pattern for arrivals, FxNS simply lets P(*n*, *t*) dictate the startup pattern.

Routing: One final issue concerns routing, i.e., selecting a next-hop router when forwarding packets. MesoNet assumes offline route computation, and expects a resulting forwarding table to be present in each router. For most routers in the topology shown in Fig. 3, next-hop forwarding is obvious, since there is a single link to each neighbor. For 16 routers (i.e., the backbone) alternate routes are possible. We computed forwarding tables with Dijkstra's Shortest-Path First (SPF) algorithm [17], using propagation delay as the metric.

Excluded MesoNet Functions: FxNS provides no implementation for MesoNet parameters: x10, x12, x13 and x14. While inducing selected spatiotemporal congestion (x13) and long-lived flows (x14) seems appropriate for experiments comparing congestion-control algorithms, such special features add little value for our experiments. Though we could include various larger transfer sizes (x12) in our experiments, we decided to omit this feature because we wanted to compare the influence of having streams of related packets (i.e., flows) against independent packet injections. If we simulated various flow sizes, then we would need to find some mapping to independent packet injection. Such a mapping would require adjusting *p* in the absence of FL to compensate for variations in average flow sizes when FL is enabled. We already needed to relate a given *p* to average flow size when FL is enabled. We determined that adding variations in average flow size would create an undesirable complication. Finally, we assumed each user has infinite patience (x10), which eliminated user patience as a factor.

We took this decision because user patience makes sense only when TCP is enabled. Since we could not formulate an analog to user patience when TCP is disabled, we decided to eliminate user patience as a factor in FxNS.

3.3.2 Dependencies among FxNS Factors

We determined implementing all seven realism elements as independent FxNS factors would prove infeasible. Instead, we identified seven dependencies among the factors, as illustrated in Fig. 4. Next we explain and justify these dependencies.

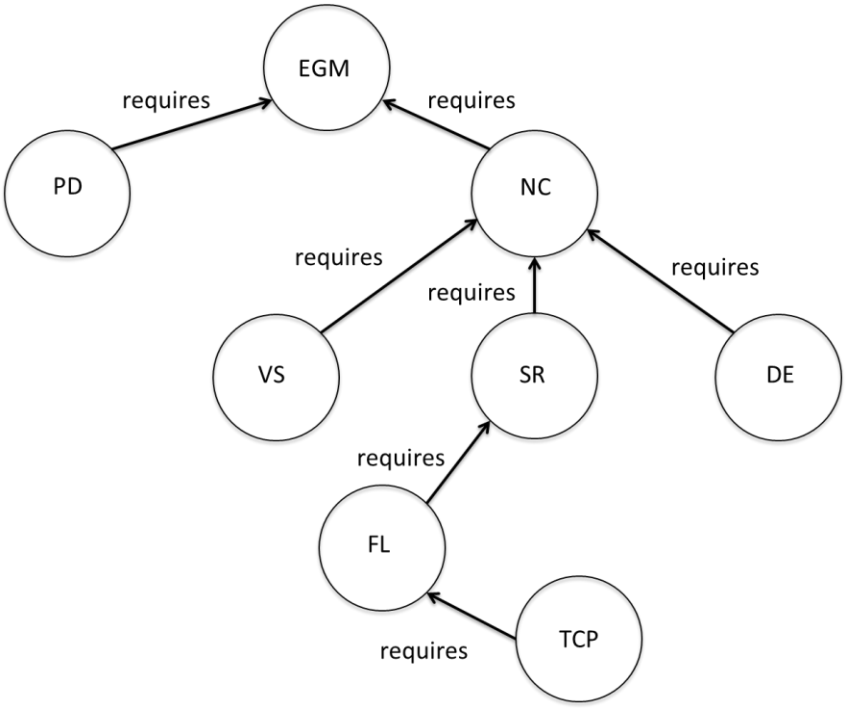

Figure 4. Dependencies among FxNS factors

The abstract EGM model is the root of our dependency tree. One can easily add finite queues (PD) to such a model. In addition, one can readily assign node classes (NC) to the topology included in such a model. Since variable speeds (VS) are assigned to nodes of different classes, node classes must be included in order to facilitate VS. Similarly, since propagation delays (DE) are assigned only to backbone links, we need to be able to distinguish such links. Backbone links connect two backbone routers, which requires that nodes be classified, creating a dependency on NC.

While sources and receivers might be included as a second tier under a flat topology, i.e., without node classification, we decided to restrict the use of sources and receivers to be a fourth tier under access routers. We took this decision for convenience, allowing us to eliminate 24 configurations that we would otherwise need to simulate. Given our decision SR requires the use of node classes.

Enabling flows (FL) considers packets injected as a stream between a source and receiver. Without the presence of SR, there would be no obvious way to identify a related stream of packets, unless we significantly complicated the packet-injection process

typically used by routers in EGM. For example, for *p* packets injected at each time step we would need to assign the packets not only to a source router but also to a "flow" within that router. To make such an assignment, a "flow" would need to exist already or else would have to be created as a new "flow." We could select a source and sink for each packet, determine if a packet traversed between them before. If so, then we could assign the packet to an existing "flow." If not, we could assign the packet to a new "flow." In either case, we would simply be implementing a packet-injection process with superfluous logic glued on. Further, such an approach would not allow us to decide when a "flow" ends. Our other option would have been to initiate a flow arrival and connection process under routers. In that case, we would need to decide how many flow arrival processes would be operating under each router. Given these issues, we simply decided that FL requires SR.

TCP regulates the rate of packet transmission within a stream of related packets, retransmitting those that are not received, and deciding when all packets in the stream have been delivered successfully. These steps cannot be taken without a flow. Thus TCP requires FL.

3.3.3 Numbering Valid FxNS Configurations

We label FxNS configurations using a numbering scheme based on binary encoding, as shown in Fig 5. Each optional factor is assigned a position in a seven-bit vector, from most (bit 7) to least (bit 1) significant. Factors are assigned to bit positions from the bottom of the dependency tree and moving upward, with TCP assigned to bit position 7 and FL to bit position 6 and so on to PD, which is assigned to bit position 1. When a selected factor is enabled its bit position is set to one, and set to zero when disabled. The resultant bit vector can be converted to a decimal value, which is the configuration number. For example, Fig. 5 shows the encoding when NC+VS+SR+FL are enabled, which translates to decimal 54. So that configuration is designated C54.

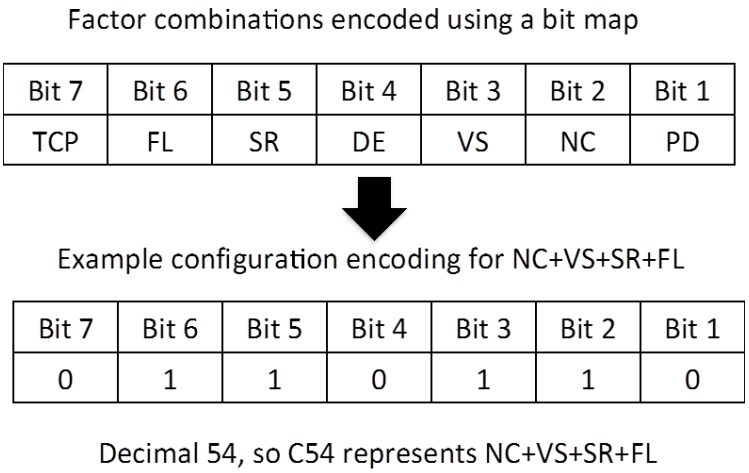

Figure 5. Numerical encoding scheme for FxNS configurations – and one example

Respecting the dependencies shown in Fig. 4, we identified 34 valid FxNS configurations. Table 4 defines those configurations and gives the configuration number, both in sequence (1-34) and in numerical encoding (C0-C127).

Table 4. Set of FxNS configurations simulated

Seq#	Config	TCP	FL	SR	DE	VS	NC	PD
1	C0	0	0	0	0	0	0	0
2	C1	0	0	0	0	0	0	1
3	C2	0	0	0	0	0	1	0
4	C3	0	0	0	0	0	1	1
5	C6	0	0	0	0	1	1	0
6	C7	0	0	0	0	1	1	1
7	C10	0	0	0	1	0	1	0
8	C11	0	0	0	1	0	1	1
9	C14	0	0	0	1	1	1	0
10	C15	0	0	0	1	1	1	1
11	C18	0	0	1	0	0	1	0
12	C19	0	0	1	0	0	1	1
13	C22	0	0	1	0	1	1	0
14	C23	0	0	1	0	1	1	1
15	C26	0	0	1	1	0	1	0
16	C27	0	0	1	1	0	1	1
17	C30	0	0	1	1	1	1	0
18	C31	0	0	1	1	1	1	1
19	C50	0	1	1	0	0	1	0
20	C51	0	1	1	0	0	1	1
21	C54	0	1	1	0	1	1	0
22	C55	0	1	1	0	1	1	1
23	C58	0	1	1	1	0	1	0
24	C59	0	1	1	1	0	1	1
25	C62	0	1	1	1	1	1	0
26	C63	0	1	1	1	1	1	1
27	C114	1	1	1	0	0	1	0
28	C115	1	1	1	0	0	1	1
29	C118	1	1	1	0	1	1	0
30	C119	1	1	1	0	1	1	1
31	C122	1	1	1	1	0	1	0
32	C123	1	1	1	1	0	1	1
33	C126	1	1	1	1	1	1	0
34	C127	1	1	1	1	1	1	1

4. Experiment Design

We designed an experiment to explore the influence of FxNS realism factors on global congestion behavior in a simulated network. Below, we first identify (Sec. 4.1) fixed input parameters used across all our simulations. Next we specify (Sec. 4.2) parameters varied from simulation to simulation. Third we define (Sec. 4.3) four responses measured for our simulations. We chose responses that can apply across all 34 configurations that we simulate. Finally, we discuss (Sec. 4.4) the fact that our simulations could individually self-adapt in two dimensions: (1) number of p values simulated and (2) number of time steps simulated for each p value.

4.1 Fixed Input Parameters

We use the same 218-node topology (recall Fig. 3) in all simulations. We adapted this topology from that of an Internet service provider. Because the topology core spans the continental United States and allows choice of routes, we used Dijkstra's SPF algorithm [17] to compute next hops for core nodes based on propagation delays. Routing to and from core nodes consists of single paths with obvious next hops. This approach creates a fixed forwarding table for each node. We used this forwarding table for all simulations. Packets are forwarded based on SPF propagation delays in the core and based on SPF hops toward and away from the core. Note that propagation delays are used to compute SPF next hops in the core regardless of whether DE is enabled or disabled. Thus disabling DE causes packets to experience no propagation delays in the core even though packets are forwarded based on SPF propagation delays.

We also fixed the measurement interval (M) to 200 time steps and executed each simulation for 1000 M. This means that each simulation executes for 200 000 ts. There might be some situations, e.g., where p is large and packet dropping (PD) is disabled, requiring excessive memory usage by our simulations. In such cases, as explained in Sec. 4.4, our simulations self-adapt to execute a lower number of time steps in order to limit memory usage.

4.2 Variable Input Parameters

We varied only two parameters: (1) packet-injection rate p and (2) configuration (as identified in Table 4). For each configuration, we varied p up to 2500 in increments of 10. This means that we planned 250 simulations for each configuration, or ($250 \times 34 =$) 8500 simulations in all. However, as explained in Sec. 4.4, simulations of individual configurations could self-adapt so that when extreme congestion appears at successive values of p then simulations terminate for the configuration. This saves computation time because once a configuration demonstrates extreme congestion for several increasing values of p then the configuration continues to exhibit congestion as p increases further.

Each configuration simulated entails a combination of enabled and disabled FxNS realism factors. For each realism factor, as shown in Table 5, we selected one set of parameter values when enabled and a different set when disabled.

Table 5. Parameter values when each FxNS realism factor is enabled or disabled

	Enabled	Disabled
PD	buffers = 250×router speed	buffers = ∞
NC	3-tier 218-node topology as in Fig. 3 – routers labeled as core, PoP, D-class, F-class or N-class	flat 218-node topology as in Fig. A2 – with router classes unlabeled
VS	core 80 p/ts; PoP 10 p/ts; D-class 10 p/ts; F-class 2 p/ts; N-class 1 p/ts; fast source/sink 2 p/ts; normal source/sink 0.2 p/ts	all routers and sources/sinks 9 p/ts
DE	core links have propagation delays	no propagation delays
SR	51 588 sources and 206 352 sinks deployed uniformly below access routers	no sources or sinks deployed
FL	transfers are packet streams: sized randomly from Pareto distribution (mean 350, shape 1.5) – streams set up with TCP connection procedures	transfers are individual packets
TCP	packet transmission regulated by TCP congestion-control including slow-start (initial *cwnd* = 2 *fsst*=100 *sst* = $2^{30}/2$) and congestion avoidance	packet transmissions not regulated by congestion-control

4.3 Responses

Given the wide range of configurations simulated in our experiment, we were constrained to choose (and define) responses comparable across all configurations: from most abstract to realistic. This prevented us, for example, from comparing behavior based on flows, since many configurations did not use flows. Despite this constraint we also desired to measure a range of behaviors, rather than limit ourselves to a single response. We determined that across all configurations two measurable concepts exist: graphs and packets. We found that we could use these simple concepts to measure: congestion spread (χ), connectivity breakdown (α), packet delivery (π) and packet latency (δ). For each configuration we simulated, we plotted each of these responses (*y* axis) against increasing packet-injection rate (*x* axis), forming a vector per configuration per response. Thus simulating each configuration results in four vectors, each defined by a set of (*x*: injection rate, *y*: response value) pairs. We define each response below.

4.3.1 Congestion Spread

To measure congestion spread, we adapt an approach from percolation theory, combined with graph theory. Our approach is inspired by the concept of giant connected

component (GCC). In graph-theory, the GCC is defined [1] as a large, connected subgraph with size *on the same order* as the number of nodes in the graph, i.e., the GCC includes *nearly all nodes*. Percolation theory [2] further mystifies the concept by stating that percolation can occur only on infinite graphs, and thus the GCC must approach an infinite number of nodes.

We take an empirical approach that replaces the GCC with the largest self-connected subgraph (LSS). We work with a finite graph of up to about a quarter million nodes. So, according to percolation theory, percolation cannot be said to occur, even when the LSS consists of all quarter million nodes. Readers should bear in mind our use of the LSS as a substitute for the GCC. Readers should also note that we consider percolation to be observed whenever the LSS contains all nodes within a topology.

General Approach: We label each node as congested, cutoff or uncongested. Here, a node is considered cutoff when it connects only to congested neighbors. We then compute self-connected subgraphs of nodes that are labeled either as congested or cutoff. We declare the largest such subgraph to be the LSS of isolated nodes. The proportion (χ) of network nodes in the LSS of isolated nodes can range between 0 (no congested nodes) and 1 (all nodes are either congested or cutoff). We plot χ to represent the spread of congestion throughout the network. Below we define the details more precisely.

Congested Node: Let $Q_{i,d}$ be the count of packets waiting for transmission in direction d (up or down or only) at node i. Node i is congested if $Q_{i,d} \geq Q_T$. Note that when NC is disabled all routers have only one queue (d=only). When NC is enabled, backbone routers have only one queue (d=only), while other router types have two queues (d=down and up). We label a router as congested if any queue within the router contains Q_T or more packets. Here, we set Q_T = 250 × forwarding speed × 0.7. Note that forwarding speed is half the router speed for PoP and access routers. Forwarding speed is the router speed for backbone routers.

Cutoff Node: Given a graph with n nodes, an adjacency matrix **A** contains rows and columns labeled with vertices (1 to n). A link exists between two nodes i and j when $a_{i,j}$ = 1 in **A**. For the same graph, a vector **C** represents the congestion state of each vertex, where a congested node is denoted by c_i = 1 and an uncongested node is denoted by c_i = 0. For any node c_i in **C** that is not congested (i.e., c_i = 0), then that node is cutoff if

$$\prod_{a_{i,j}=1} c_j = 1 \tag{3}$$

i.e., all its surrounding neighbors (not including sources and receivers) are congested.

Largest Self-Connected Subgraph (LSS) of isolated nodes: We define the LSS of nodes that are congested or cutoff, including any sources and receivers under such nodes, as the LSS of isolated nodes. Effectively, if sources and receivers connect to the network through a congested or cutoff node, then we also label those sources and receivers as isolated nodes.

Proportion of nodes in the LSS of isolated nodes: Let $\mathbf{G}_N$ be the set of all nodes in a graph **G**, let $\mathbf{G}_\chi$ be the set of nodes in the LSS of isolated nodes in **G**, where $\mathbf{G}_\chi \subseteq \mathbf{G}_N$. Then $\chi = |\mathbf{G}_\chi|/|\mathbf{G}_N|$ is the proportion of nodes in **G** that are in the LSS of isolated nodes.

Proportion of nodes in the LSS of congested nodes: In Appendix B, we consider the LSS for congested nodes (γ) only. Let $\mathbf{G}_\gamma$ be the set of nodes in the LSS of congested nodes in $\mathbf{G}$, where $\mathbf{G}_\gamma \subseteq \mathbf{G}_N$. Then $\gamma = |\mathbf{G}_\gamma|/|\mathbf{G}_N|$ is the proportion of nodes in $\mathbf{G}$ that are in the LSS of congested nodes. As shown in Appendix B, the results for γ are similar to the results for χ.

4.3.2 Connectivity Breakdown

We use a similar approach to measure the breakdown of network connectivity. Here we label nodes as congested, cutoff or uncongested, using the definitions given above, and then we compute self-connected subgraphs of nodes that are labeled as uncongested. We declare the largest such subgraph to be the LSS of reachable nodes. The proportion (α) of network nodes in the LSS of reachable nodes can range between 0 (no uncongested nodes) and 1 (all nodes uncongested). We plot α to represent the breakdown in network connectivity as α falls from 1 to 0. While in some cases α is the inverse of χ, an inverse relationship does not always hold. For that reason we report both α and χ. Assuming the definitions given above, we define residual details below.

Largest Self-Connected Subgraph (LSS) of reachable nodes: We define the largest self-connected subgraph of nodes that are uncongested, including any sources and receivers under such nodes, as the LSS of reachable nodes. The network is fully connected when this GCC contains all nodes in the network.

Proportion of nodes in the LSS of reachable nodes: Let $\mathbf{G}_N$ be the set of all nodes in a graph $\mathbf{G}$, let $\mathbf{G}_\alpha$ be the set of nodes in the LSS of reachable nodes in $\mathbf{G}$, where $\mathbf{G}_\alpha \subseteq \mathbf{G}_N$. Then $\alpha = |\mathbf{G}_\alpha|/|\mathbf{G}_N|$ is the proportion of nodes in $\mathbf{G}$ that are in the LSS of reachable nodes.

4.3.3 Packets Delivered

Packets injected into a network will meet one of three fates: be delivered, be queued, or be discarded. We consider a network as effective when all packets are delivered and useless when no packets are delivered. Based on this reasoning, we measure the proportion (π) of injected packets that are delivered during a simulation. The proportion of packets delivered (π) can range between 0 (no packets delivered) and 1 (all packets delivered). Next, we more precisely define this measure.

Let a_e be the aggregate number of packets injected into the network over the course of a simulation, i.e., over the time span $t = 1 \ldots e$, as defined by eq. 4.

$$a_e = \sum_{t=1}^{e} p_t \tag{4}$$

Let q_e be the aggregate number of packets queued in all buffers over all network nodes at time step e, i.e., the end of a simulation, as defined by eq. 5.

$$q_e = \sum_{\substack{i=1..n \\ d=\text{all}}} Q_{i,d} \tag{5}$$

Let $b_{i,d}$ be the maximum number of packets that can be queued in node i for transmission in direction d. When packet dropping is not used $b_{i,d} = \infty$. For a given node i, an arriving

packet will be dropped when, for any direction d, $Q_{i,d} = b_{i,d}$, i.e., there is no room for the packet. Let b_e denote the aggregate number of packets dropped by all network nodes from all queues over the course of a simulation, i.e., over the time span t = 1…e.

The proportion of packets queued (ρ) then is $\rho = q_e/a_e$, and the proportion of packets dropped (x) is $x = b_e/a_e$. The proportion of packets delivered (π) then is $\pi = 1 - \rho - x$. This estimates the probability that a packet will be delivered. When PD is disabled $x = 0$. When PD is enabled, $\rho \rightarrow \varepsilon$, where ε is a small, fixed, upper bound, established by the aggregation of buffer sizes across all nodes. Thus for disabled PD $\pi = 1 - \rho$, and for enabled PD $\pi = 1 - \varepsilon - x$. This implies that $1 - \rho = 1 - \varepsilon - x$, and so $\rho = x + \varepsilon$. Since ε is relatively small, $x \approx \rho$. So when PD is enabled, π is driven by x; otherwise by ρ. So regardless of the state of PD, π reasonably measures effectiveness of packet delivery.

4.3.4 Packet Latency

While a network that delivers a high proportion of injected packets can be considered effective, that same network can be considered inefficient if excessive time is required to move packets from point of injection to point of extraction. To assess efficiency for delivered packets (i.e., packets in π) we measure the average one-way delay (Δ), which will be longer as queues are larger and shorter as queues are smaller. As we describe below, we scale Δ to be average delay (δ) within the interval [0...1]. Next we precisely define Δ and its scaling to δ.

Let P_π be the set of packets injected into the network that reach the intended destination. Let s_i be the creation time of the ith packet and let d_i be the delivery time of ith packet. Then average one-way packet delay (Δ) is defined by eq. 6.

$$\Delta = \frac{\sum_{i \in P_\pi} d_i - s_i}{|P_\pi|} \tag{6}$$

Let $\Delta_{c,p}$ be the average one-way delay for configuration c and packet injection rate p. Let Δ_{MIN} be the minimum $\Delta_{c,p}$ over all configurations and packet injection rates, and Δ_{MAX} be the maximum $\Delta_{c,p}$ over all configurations and packet injection rates. Then scaled average delay (δ) is defined by eq. 7.

$$\delta = (\Delta - \Delta_{MIN}) / (\Delta_{MAX} - \Delta_{MIN}) \tag{7}$$

4.4 Simulation Self-adaptations

Simulating 34 configurations over 250 packet-injection rates (p), where each simulation covers 200 000 time steps, requires significant computation. In addition, simulations without packet dropping can require excessive memory usage, especially at higher values of p. To address these issues, we implemented two forms of self-adaptation within FxNS: congestion self-adaption and time-step self-adaptation.

First, we allowed FxNS to examine the history of simulated congestion spread. For a given configuration, when congestion spreads through all nodes for three successive values of p then FxNS terminates simulations for the configuration. In our experiment, the earliest any configuration terminated under congestion self-adaption was

after *p* passed 790. This saved computation time because once a configuration demonstrates extreme congestion for several increasing values of *p* then the configuration will continue to exhibit congestion as *p* increases further. Curtailing simulation of heavily congesting configurations did not cause information loss, as response variables for higher values of *p* can be extrapolated easily.

Second, we allowed FxNS to examine memory usage by individual simulation runs. When memory usage exceeded a specified threshold, subsequent FxNS simulation runs for the same configuration terminated prior to executing all 200 000 time steps. But FxNS continued to simulate subsequent values of *p*. Without time-step self-adaptation FxNS simulations could consume too much memory. In our experiment, time-step self-adaptation occurred in configurations with PD disabled, and for values of *p* greater than 250. No simulation executed fewer than 41 400 time steps. Time-step self-adaptation was triggered only in cases where packet queues were large, and so congestion extreme. Curtailing a data point early under such conditions did not lead to significant information loss, as affected configurations had already congested sufficiently to yield insightful results.

5. Results

Below we plot results for each of the four responses: congestion spread (χ), breakdown in network connectivity (α), and both effectiveness (π) and efficiency (δ) of packet delivery. Each result covers 34 configurations with *p* ranging up to 2500. Here, we plot together all 34 configurations for each response. This enables comparison of similarities and differences among configurations. Larger plots for each response-configuration pair are available elsewhere [18].

Figure 6 shows 34 plots of χ (*y* axis) vs. *p* (*x* axis), one for each FxNS configuration. These plots show how much congestion spreads in the network for the given combinations of realism factors. Figure 7 shows similar plots, but for α. These plots show how connectivity breaks down in the network for given combinations of realism factors. Figure 8 shows the proportion of packets delivered (π) varying with *p* for each of the 34 FxNS configurations. Figure 9 shows the scaled average one-way delay (δ) of delivered packets varying with *p* for each of the 34 FxNS configurations.

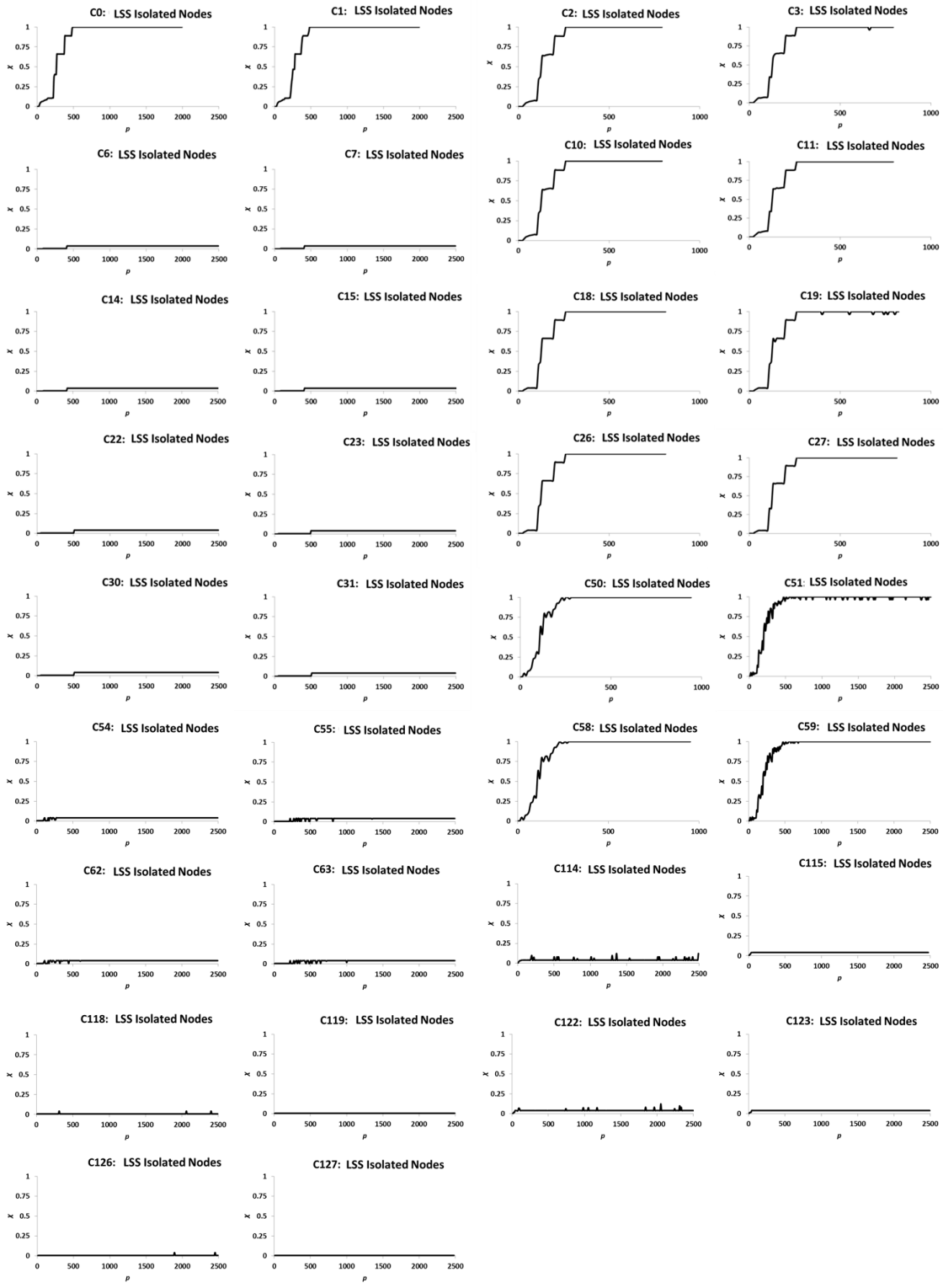

Figure 6. Proportion of nodes in LSS of isolated (χ) nodes for 34 FxNS configurations

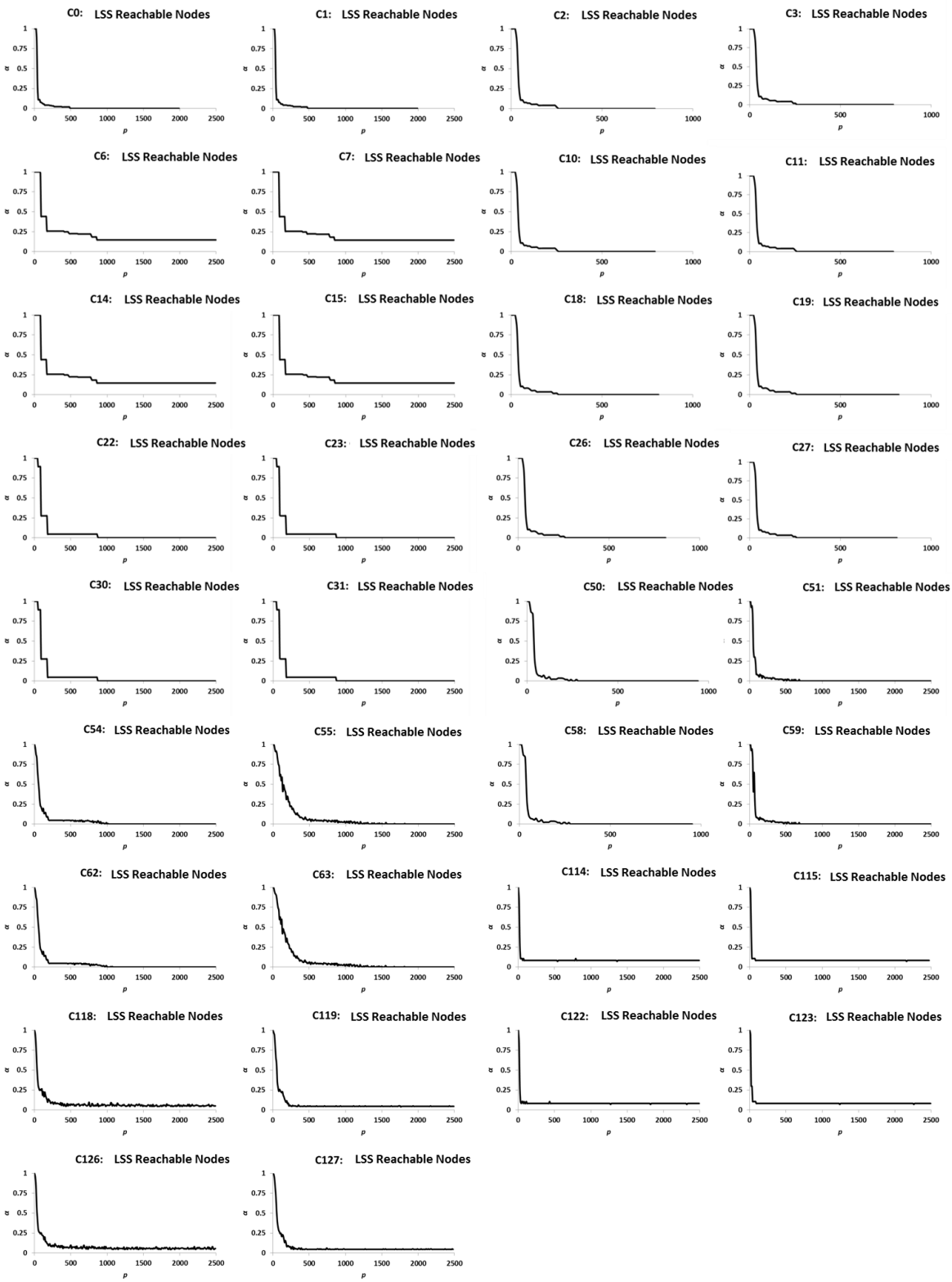

Figure 7. Proportion of nodes in LSS of reachable nodes (α) for 34 FxNS configurations

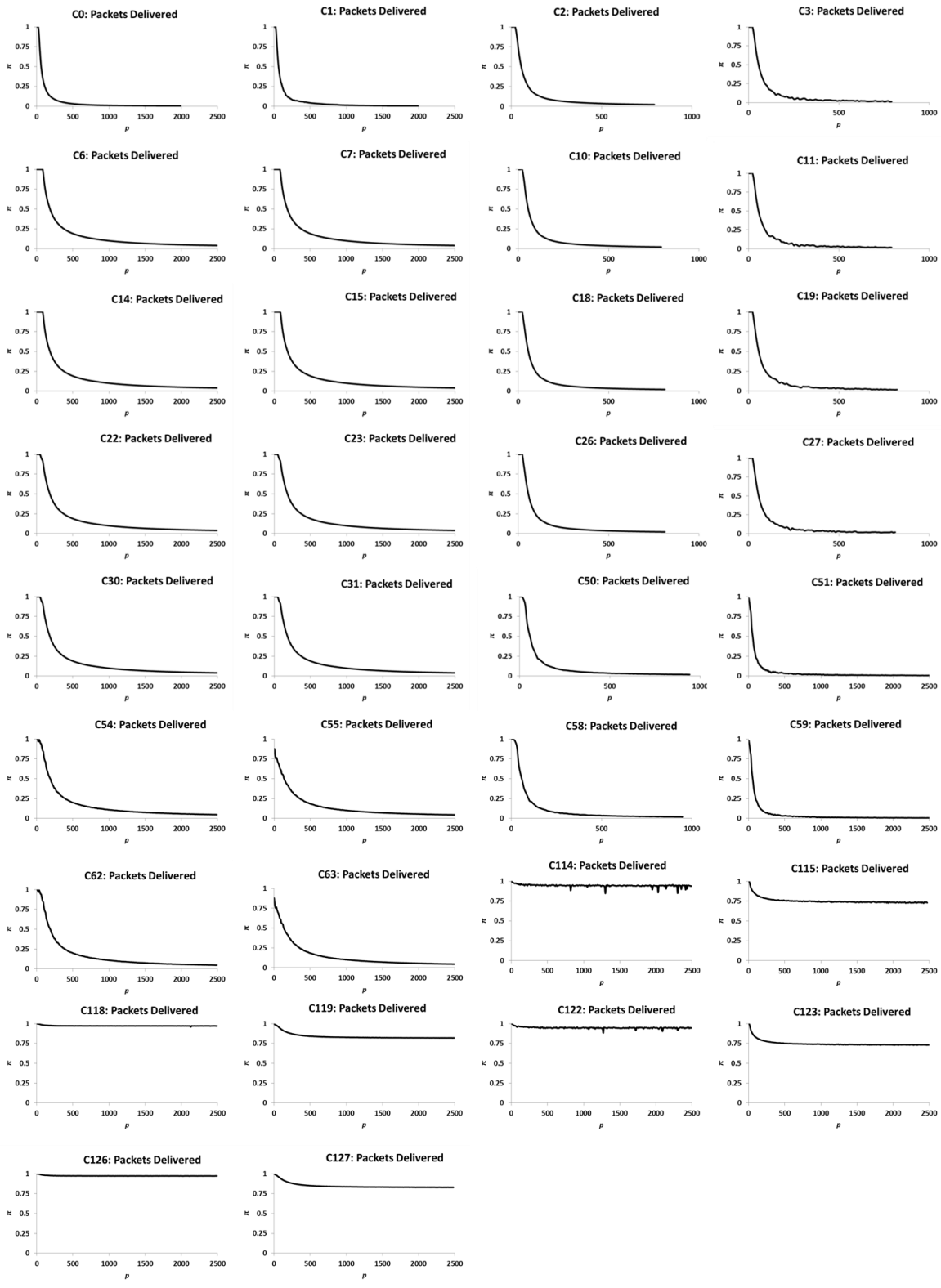

Figure 8. Proportion of packets delivered (π) for 34 FxNS configurations

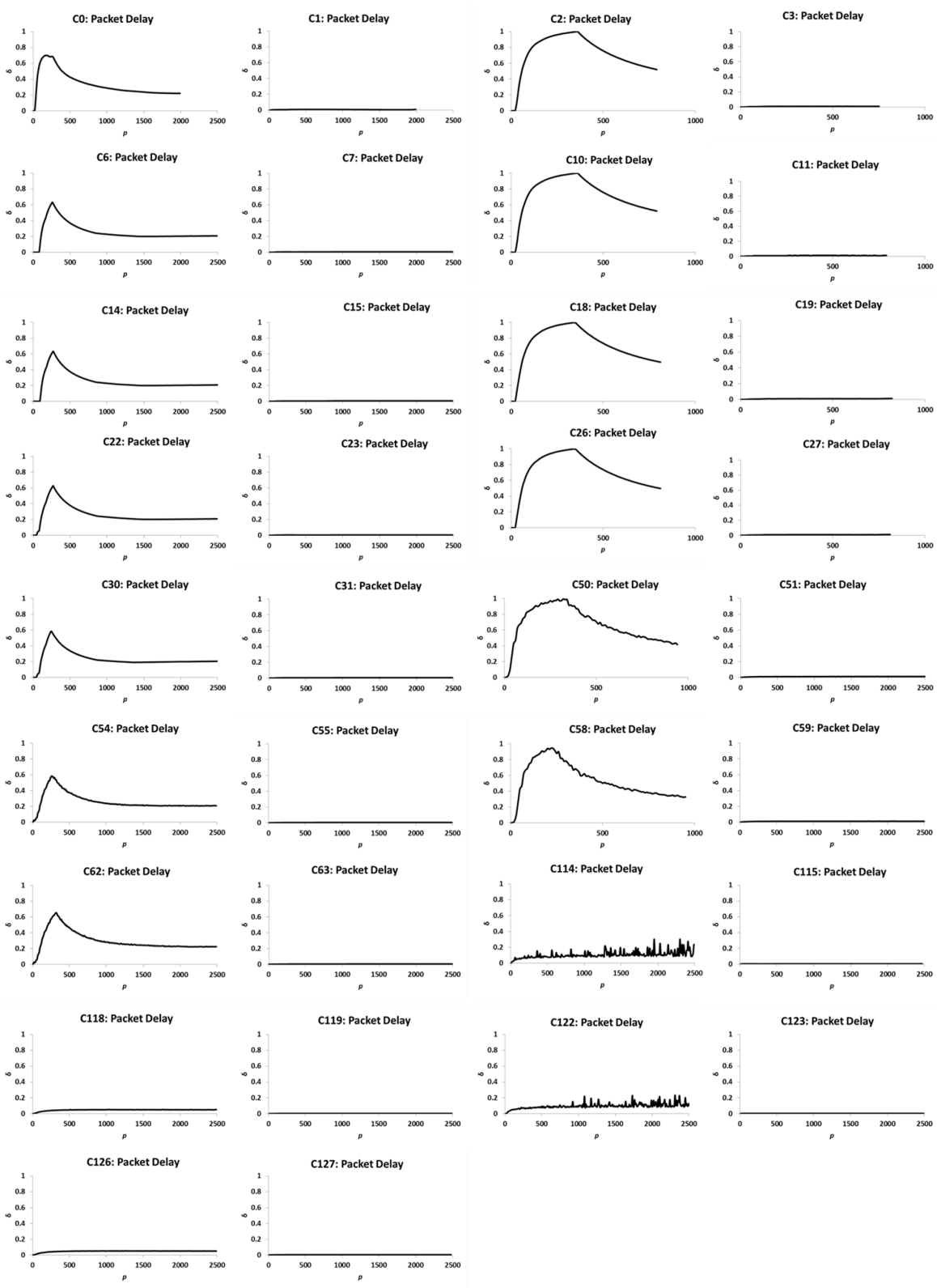

Figure 9. Scaled average latency of delivered packets (δ) for 34 FxNS configurations

6. Discussion

The plots shown in Figs. 6-9 illustrate differences in global congestion behavior among various simulated FxNS configurations. The plots also suggest that patterns of similarity exist among subsets of the configurations. Here, we explore these differences and similarities, aiming to draw some conclusions about the influence of realism on congestion in network simulations. We begin, in Sec. 6.1, by comparing plots between the most abstract (C0) and realistic (C127) configurations.

Subsequently, for each response, we examine similarities and differences among all configurations. We do this by transforming each of the 34 plots for a given response to a vector and then hierarchically clustering configurations based on the squared Euclidean distances between the vectors. Some of the vectors are shorter than others because FxNS implements congestion self-adaptation, which terminated affected configurations before reaching $p = 2500$. In such cases, we filled the missing vector elements with extrapolated results. For the LSS of isolated nodes, discussed in Sec. 6.2, we filled the missing elements with ones. For the LSS of reachable nodes, discussed in Sec. 6.3, and the proportion of packets delivered, discussed in Sec. 6.4, we filled the missing elements with zeros. For average one-way delay of delivered packets, discussed in Sec. 6.5, we filled the missing elements with data points along a linear trend line. We summarize our overall findings in Sec. 6.6.

6.1 Most Abstract vs. Most Realistic

Figure 10 contains four subplots comparing congestion behavior between the most abstract (C0) and realistic (C127) configurations. The subplots compare: congestion spread (χ), connectivity breakdown (α), packets delivered (π), and latency (δ) for delivered packets. We restrict the subplots to the range $p \leq 2000$ because FxNS terminated simulations of configuration C0 early, due to congestion self-adaptation. We discuss each subplot in turn.

Congestion Spread (χ): For configuration C0 congestion spreads quickly with increasing packet-injection rate, encompassing all nodes by the time p reaches 500. For configuration C127, congestion spread remains low over the entire range of packet injection rates, even out to $p = 2500$ (not shown). This difference has two main causes. First, all nodes in configuration C0 operate at the same speed. This means that backbone nodes become overwhelmed with congestion, which then spreads outward to the network edge. In configuration C127 router nodes are engineered with varying, hierarchical speeds, so higher tiers can handle the packet inflow rate from lower tiers. Second, configuration C0 does not monitor and adapt to congestion, while configuration C127, which implements TCP, measures congestion and adapts the rate of packet inflow accordingly.

Connectivity Breakdown (α): Network connectivity breaks down rather quickly for both configuration C0 and C127, reaching a relatively low level before p reaches 500. Even so, there are two main differences in the subplot: C127 decays more slowly than C0 and C127 asymptotes at a higher level of network connectivity than C0, which drops to zero after p passes 500. C127 decays more slowly because TCP adapts packet injection based on measured congestion. C127 asymptotes at a higher level because variable router

speeds restrict congestion to the network edge. The network core remains uncongested and intact. Connectivity breaks down completely for C0 because the network core becomes congested and then congestion spreads to the edge, consuming all nodes.

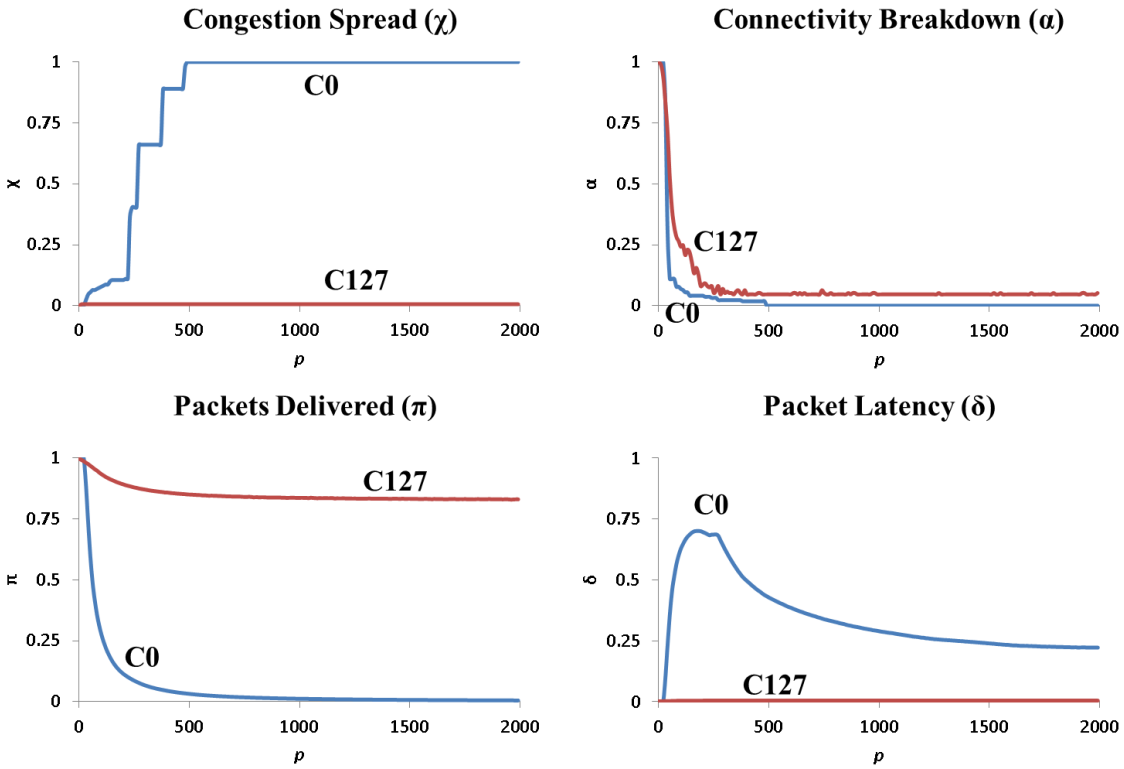

Figure 10. Comparison of congestion spread (χ), connectivity breakdown (α), packets delivered (π), and packet latency (δ) for the most abstract (C0) and most realistic (C127) FxNS configurations

Packets Delivered (π): For C0 the proportion of packets delivered drops steeply, reaching nearly zero as *p* passes 1000. For C127 the proportion of packets delivered drops only modestly with increasing *p*, stabilizing near 80 %. This large difference arises from a combination of two factors: TCP and packet dropping. C0 does not adapt packet injection based on measured congestion and does not discard packets. With increasing *p*, this leads to a large and growing backlog of packets in all network nodes. C127, which implements TCP and packet dropping, adapts packet injection based on measured congestion and also discards packets when router buffers fill. As a result, undelivered packets for C127 encompass those that are discarded, and the number of packets that must be discarded is limited by the rate adaptation of TCP.

Packet Latency (δ): For C127 the latency of delivered packets remains low even as *p* increases to and beyond 2000. This occurs because packet dropping limits the size of router queues, so delivered packets are not delayed very long. For C0, which does not implement packet dropping, packet latency climbs steeply with increasing *p*, reaching an apex before decaying gradually. The reason for the steep climb is that packet queues become jammed, which drives up packet latency. The reason for the gradual decay is that latencies are recorded only for delivered packets. At high values of *p*, C0 delivers

relatively few packets, and those packets necessarily transit routes where queues are not jammed. Even with such decay, packet latency for C0 remains significantly higher than for C127.

The foregoing discussion contains explanations for the differences in the subplots in Fig. 10. Some of those explanations are informed by results from the clustering analyses that follow in Secs. 6.2-6.5. Other explanations, relating to the progression of congestion within the network topology, are informed by a visualization of the spread of congestion. FxNS can export the ending congestion state for each configuration and simulated value of *p*. Phillip Gough, a researcher from Australia's Commonwealth Scientific and Industrial Research Organization, devised and implemented an interactive, multidimensional visualization that allowed us to examine the spread of congestion in the network topology for all configurations and values of *p*. Figure 11 shows a screenshot from that visualization. An interactive version, using data from our experiment, is available elsewhere [19].

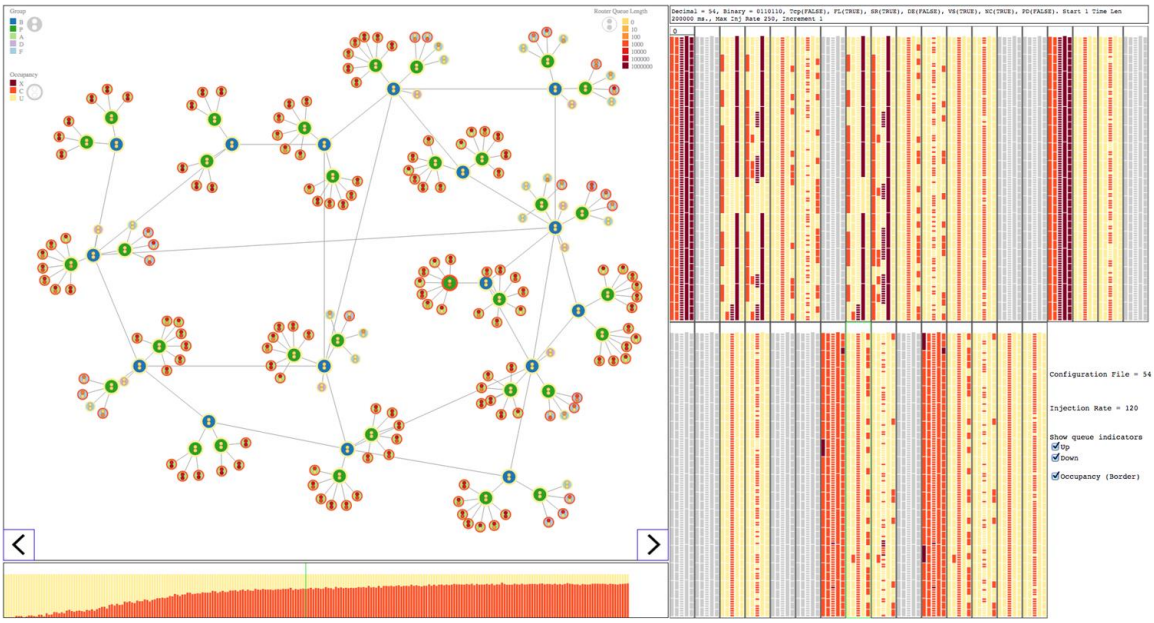

Figure 11. Screenshot from dynamic visualization (courtesy Phillip Gough) of node status with increasing *p* for 34 FxNS configurations in a 218-node topology

The visualization contains three main windows. The network topology is shown in the upper left-hand window. Routers are shown as circles and links as line segments between circles. Router classes (backbone, point-of-presence and D-, F- and N-class access) are distinguished by color. Occupancy status (congested, cutoff and uncongested) is indicated by a colored outline around each router. Two smaller, but optional, circles within each router represent queue sizes, coded by color. Color keys are provided in the visualization.

Immediately below the topology is a stacked bar graph, where each cell represents a specific value of *p*, from 1 (left) to an upper bound (right). Each cell also indicates the proportion of routers that are uncongested (light color), congested (medium color) and cutoff (dark color) for the relevant value of *p*. The right-hand window shows a strip for each of the 34 FxNS configurations. Each strip contains 218 cells, each representing a

router. The cells are arranged in vertical columns, where each column displays routers in a particular class. Each cell is color coded with the occupancy status of the related router at a selected value of *p*.

The user can select a configuration in the right-hand window. The selected configuration is indicated by a green dot. Selecting a specific configuration changes the color coding of the topology and the related stacked bar graph to match the congestion state for the configuration. Selecting a cell on the stacked bar graph adjusts the congestion state of the topology to the related value of *p*. Selecting a *p* value also adjusts the congestion state of all configurations on the right-hand window to reflect that *p* value. The user can step through various *p* values, while watching changes in congestion within the displayed topology. Alternatively, the user can step through various *p* values while using the right-hand window to compare congestion patterns among all configurations. These are the tools that allowed us to explain some differences (above) when comparing congestion evolution between configurations C0 and C127. These tools also allowed us to see similarities and differences among congestion patterns in various configurations.

6.2 Congestion Spread

Figure 12 shows a hierarchical clustering among the vectors in Fig. 6, which plotted the proportion of nodes in the LSS of isolated nodes for the 34 FxNS configurations. The *x*-axis of Fig. 12 is labeled with sequential configuration numbers from Table 4. The *y*-axis reports squared Euclidean distance between the vectors comprising each configuration.

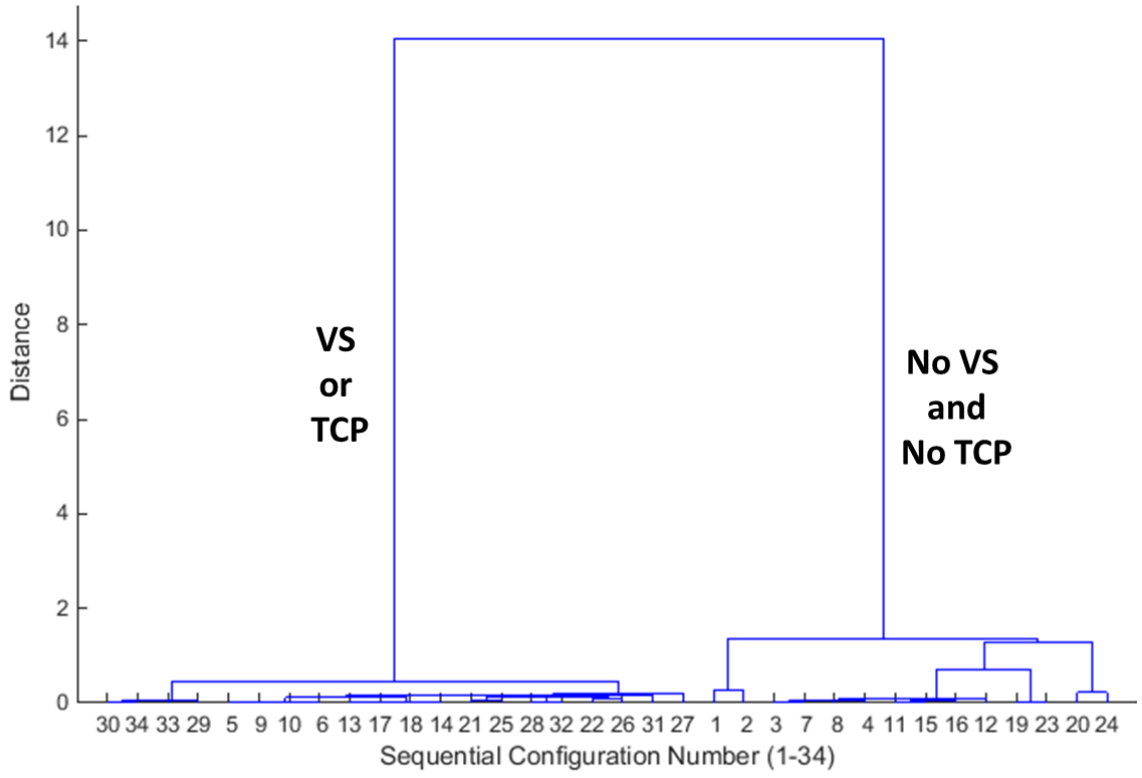

Figure 12. Clustering of LSS of isolated nodes (χ) based on Squared Euclidean Distance

The clustering plot indicates two main groupings, separated by a large distance. The left-hand group contains configurations that enabled variable router speeds (VS) or TCP or both. Configurations in this group correspond to those that show little congestion spread. The right-hand group contains configurations that did not enable VS and did not enable TCP. Configurations in this group correspond to those that showed congestion spreading throughout the network topology. Note the most abstract configuration, C0 (sequence number 1), appears in the congested group, while the most realistic configuration, C127 (sequence number 34), appears in the uncongested group. The reasons why VS and TCP have these effects were explained above in Sec. 6.1. Most network models surveyed [5-14] in Sec. 2 are quite similar to configuration C0, while real networks are modeled [15-16] more like configuration C127. This is evidence that many previous studies report congestion spread unlikely to appear in real networks.

6.3 Connectivity Breakdown

Figure 13 shows a hierarchical clustering among vectors in Fig. 7, which plotted the proportion of nodes in the LSS of reachable nodes for the 34 FxNS configurations. Note that distances among the clusters in Fig. 13 are much smaller than those seen among clusters in Fig. 12. This reflects the fact that breakdown in network connectivity is more similar among configurations than congestion spread. Breakdown in network connectivity occurs in cases where subgraphs of the topology are disconnected (due to congestion) from other subgraphs. As load increases network connectivity breaks down even in cases where congestion does not necessarily spread widely. The studies we surveyed [5-14] address spreading network congestion but do not address breakdown of network connectivity. This aspect of network congestion seems important and should be examined in future studies.

We labeled Fig. 13 to indicate factors in common among various groupings of FxNS configurations. While the groupings are not as clear as those shown in Fig. 12, our labeling reflects presence and absence of VS and TCP, which have significant influence on breakdown in network connectivity. Note that C0 falls into a grouping with VS disabled, while C127 falls into a grouping with TCP enabled. From the earlier discussion (Sec. 6.1) recall that TCP slows breakdown in network connectivity. Also recall that, when coupled with TCP, VS ensures the network core remains uncongested and intact.

Among configurations with VS disabled, the leftmost grouping of configurations (sequence numbers 3, 4, 7, 8, 11, 12, 15 and 16) in Fig. 13 reach complete breakdown in network connectivity sooner than other configurations with VS disabled. These configurations have NC enabled, which means that packet injection occurs at the network edge, thus packets flow in a concentrated fashion to and through the network core. This differs from configurations C0 and C1 (sequence numbers 1 and 2), where packet injection can occur at any node, thus fewer packets flow across the network core. Sequence numbers 19, 20, 23 and 24 represent configurations with sources and receivers and flows enabled. These configurations reach complete breakdown in connectivity later than those in the leftmost group. Sequence numbers 20 and 24 represent configurations with packet dropping enabled. Here, complete breakdown in connectivity is postponed

beyond that for 19 and 23. Overall, most configurations with VS disabled lost network connectivity quickly and completely.

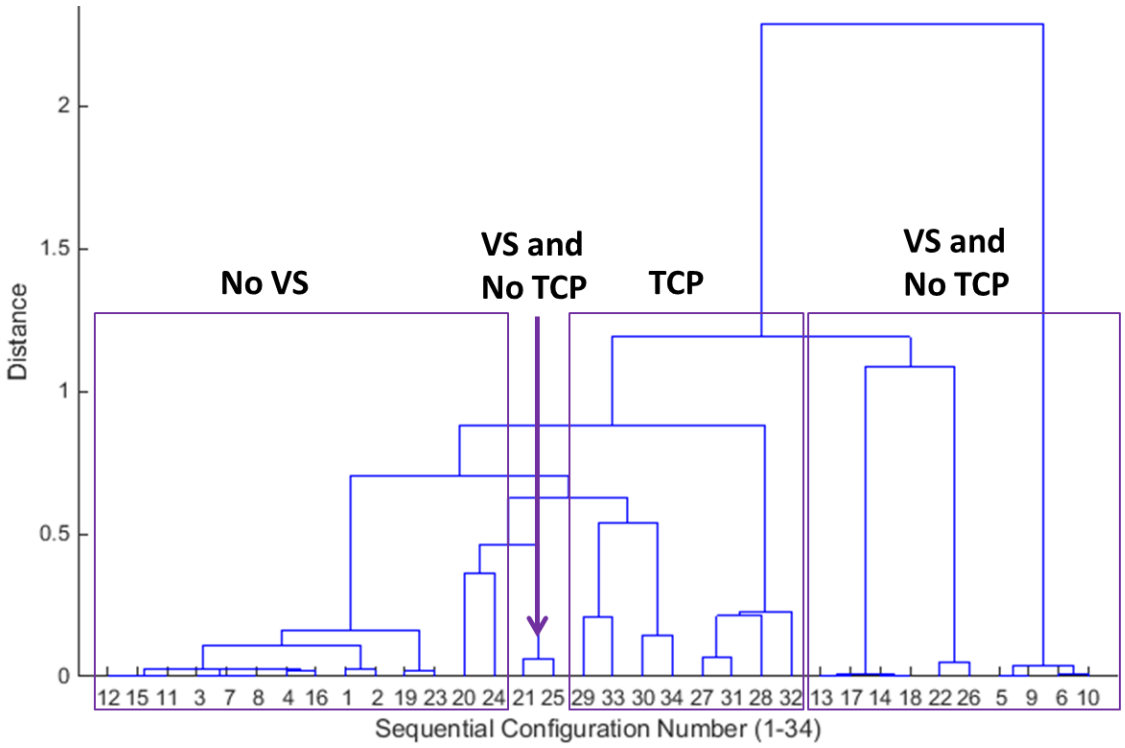

Figure 13. Clustering of LSS reachable nodes (α) based on Squared Euclidean Distance

Configurations with VS enabled but with TCP disabled can also experience complete breakdown in network connectivity, but the process takes somewhat higher packet-injection rates because more pressure must be applied from the network edges before the core can congest. Such behavior arises for configurations (C22, C23, C30, C31, C54, C55, C62, and C63) with SR (sources and receivers) enabled. Enabling SR allows more pressure to be applied from the network edge because more potential packet-injection sources reside there. When flows are enabled the decay in network connectivity slows somewhat. When further enabling packet dropping, the decay slows even more.

In configurations that enable VS and disable TCP and SR (e.g., C6, C7, C14 and C15), the network core remains uncongested and intact. With TCP enabled and VS disabled (C114, C115, C122 and C123), congestion builds in the core and oscillates in PoP routers, but the access routers remain uncongested. With both TCP and VS enabled (C118, C119, C126 and C127), congestion stays mainly within access routers, oscillating inward toward PoP routers. Overall, configurations with VS enabled retained connectivity in the network core. These results provide evidence that VS plays a key role in limiting the breakdown of network connectivity to the edge, allowing the network core to remain uncongested and intact.

6.4 Packets Delivered

Figure 14 shows a hierarchical clustering among the vectors in Fig. 8, which plotted the proportion of packets delivered for the 34 FxNS configurations. The clustering plot indicates two main groupings, separated by a large distance. The leftmost group contains configurations with TCP disabled, while the rightmost group contains configurations with TCP enabled. TCP measures the congestion state of the network and adapts packet-injection rate accordingly. This improves significantly the likelihood that an injected packet will reach its intended destination. Disabling TCP increases the likelihood that an injected packet will be queued or discarded.

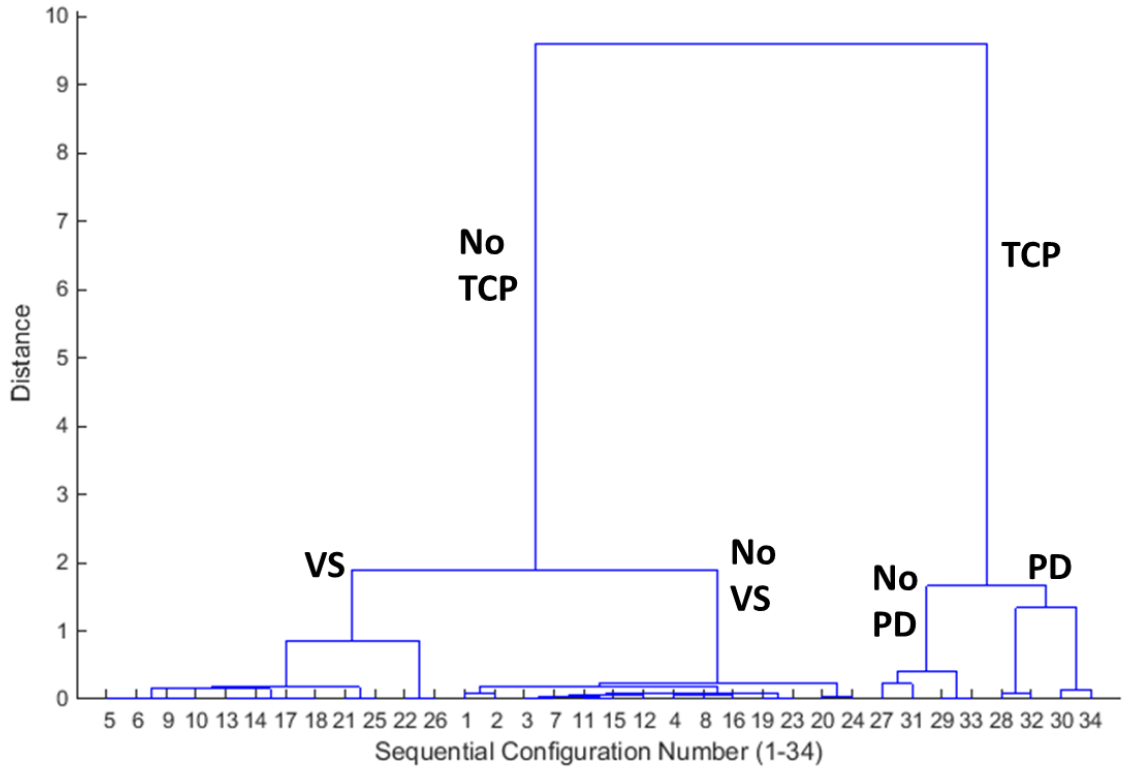

Figure 14. Clustering of packet delivery effectiveness (π) based on Squared Euclidean Distance

When TCP is enabled packet dropping (PD) has a secondary influence on the likelihood of packet delivery. Disabling packet dropping ensures that injected packets are never discarded, thus packets will eventually be delivered successfully. The buildup of packet queues, though, can delay delivery of data and acknowledgment packets, leading to timeouts and subsequently to lower throughput, as TCP significantly reduces packet-injection rate. Enabling packet dropping means that some packets will be discarded because router buffers are full. Our results found about a 20 % loss rate at high loads. With packet dropping enabled, TCP does not need to reduce as significantly the packet-injection rate, thus throughputs remain higher, while the likelihood of delivery decreases.

When TCP is disabled variable router speed (VS) has a secondary influence on likelihood of packet delivery. Absence of VS allows packet queues to build more widely

among nodes throughout a network topology. This means that packets are more likely to be queued or discarded (depending on the setting for packet dropping) when they arrive at a router. Where packets are queued, the queue length of any router is likely to be long and packet-delivery delays increase significantly. In either case, the likelihood of packet delivery quickly approaches zero. When VS is enabled packet queues build in access routers at the network edge. This reduces the number of nodes in the network where packets can be dropped or queued. In such cases, the likelihood of packet delivery approaches zero at a slower rate.

6.5 Packet Latency

Figure 15 shows a hierarchical clustering among vectors in Fig. 9, which plotted scaled average latency of delivered packets for the 34 FxNS configurations. We labeled Fig. 15 to indicate factors in common among various groupings of FxNS configurations. The leftmost half of the configurations have packet dropping (PD) enabled, while the rightmost half have PD disabled. When PD is enabled, successfully delivered packets experience very little queuing delay, thus the average latency is quite low. When PD is disabled packet queues can become quite large with increasing load, thus average latency increases. As discussed earlier in Sec. 6.1, the increase in delay reaches an apex and then declines gradually because only successfully delivered packets have one-way delays. At higher loads, successfully delivered packets experience smaller queues, or else they could not be successfully delivered during the simulation duration.

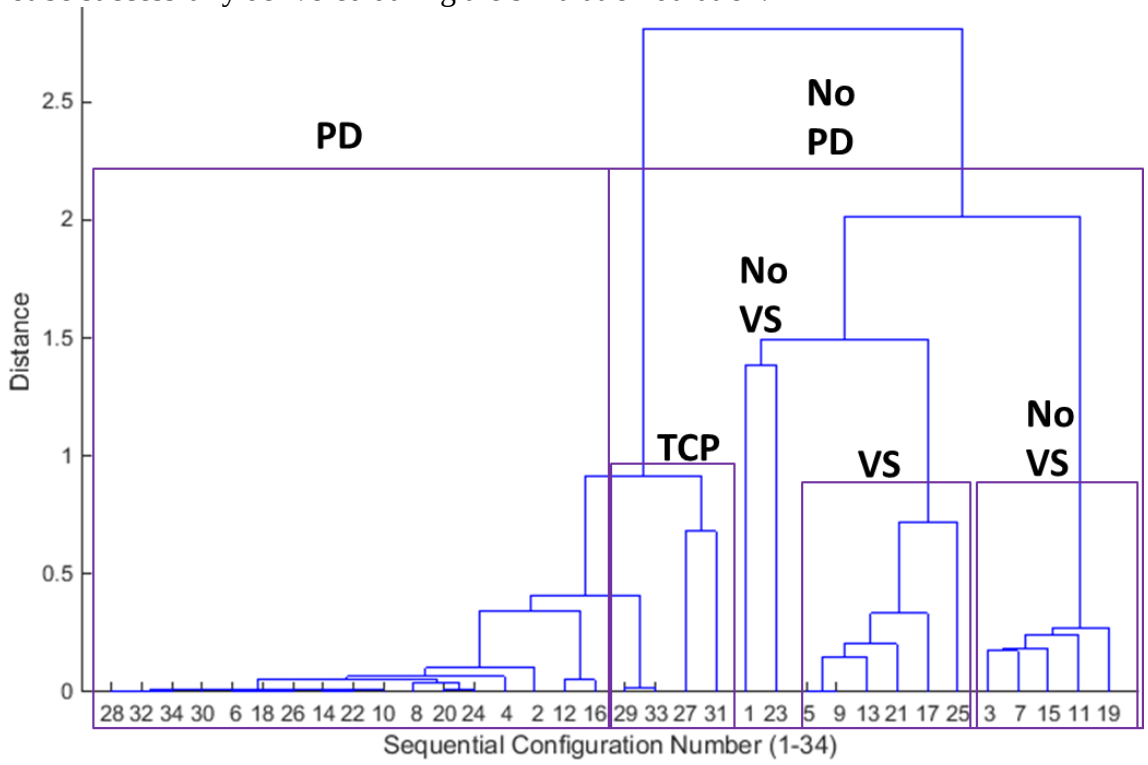

Figure 15. Clustering of packet delivery efficiency (δ) based on Squared Euclidean Distance

Some secondary factors influence packet latency when PD is disabled. Enabling TCP allows rate adaptation, thus buildup of large queues is less likely, as TCP slows packet injection when congestion appears. This significantly reduces delay experienced by successfully delivered packets. Enabling VS restricts large packet queues to routers at the network edge, which means that successfully delivered packets will have fewer large queues to transit through. Disabling VS allows large packet queues to form at any router in the topology, which means that successfully delivered packets will have to transit through more large queues.

6.6 Overall Findings

Our abstract and realistic network models exhibited very different congestion behaviors. Under increasing load, the abstract model (C0) congested quickly and completely, while the realistic model (C127) did not exhibit widespread congestion even under heavy load. The realistic model exhibited congestion only at network edges, while the backbone remained uncongested and intact. Both the abstract and realistic network models lost connectivity quickly under increasing load, but the realistic model lost connectivity less rapidly. Congestion in the abstract model spread from the network core toward the edges, leading to zero reachable nodes, while the realistic model ensured that nodes in the network core remained reachable. Further, increasing load led the abstract model to successfully deliver below 1 % of injected packets, while the realistic model delivered about 80 % of injected packets, even under high loads. Similarly, increasing load caused packet delays to spike quickly for the abstract model, while packet delays remained very low for the realistic model. Based on these findings, we conclude that the decade of studies we surveyed [5-14] cannot be relied upon as guides to congestion spread in the Internet. We reach this conclusion because the studies contain models very similar to our abstract model.

Our clustering analyses show the critical importance of modeling TCP, the congestion-control protocol used in over 90 % of flows transiting the modern Internet. In our study, congestion monitoring and rate adaptation provided by TCP was responsible for limiting the spread of congestion and the breakdown in network connectivity, especially when combined with correctly modeled variable router speeds. The rate adaptation provided by TCP was also a primary factor to increase successful delivery of packets under increasing network load. The rate adaptation provided by TCP was a secondary factor ensuring low latency among successfully delivered packets. None of the studies we surveyed [5-14] modeled TCP.

Our clustering analyses show the critical importance of modeling variable speeds among router tiers, engineered to ensure that higher tiers provide adequate throughput at the maximum possible input rate from lower tier routers. Modeling variable router speeds was a primary factor responsible for accurately simulating congestion spread and the breakdown of network connectivity. In addition, modeling variable router speeds had secondary influence on the degree of successful packet delivery and the latency for successfully delivered packets. Only one [11] study we surveyed modeled variable speeds among routers. Even in that case router speeds were not varied hierarchically.

Our clustering analyses show the importance of modeling packet dropping in order to obtain accurate measures of packet latency. Packet dropping ensures that FIFO buffers in routers limit queuing delays experienced by successfully delivered packets. Three [6, 10, 13] of the studies we surveyed modeled finite buffers, but one [10] modeled buffers as last-in first-out, another [6] used full buffers only to restrict packet injections rather than to drop arriving packets, and the other [13] discards the oldest packet to make room for the newest. Most of the studies [5, 7, 8, 11, 12, 14] we surveyed assumed infinite packet queues and then measured the resulting buildup of packets in the network as a signal of rising congestion. The real Internet uses finite, FIFO, drop-tail buffers, which discard packets arriving at full queues. None of the studies we surveyed used finite FIFO drop-tail queues.

Our clustering analyses showed propagation delays in the backbone were unimportant to model. While this appears true for networks spanning the continental United States (as ours did), propagation delays could become important when modeling a global network or a network containing links transiting satellite hops. Certainly, propagation delay would be important to model when considering inter-planetary networks. In our model, delays due to queuing dominated delays due to propagation. In reverse situations, propagation delay would be important to model.

A decade of simulation studies [5-14] investigated congestion spread in network topologies, often finding that congestion can be modeled as a percolation process on a graph, spreading slowly under increasing load until a critical point, after which congestion spreads astonishingly quickly throughout the entire network. Those same studies identified various measurable signals that arise around the critical point. Such signals might facilitate prediction of the onset of widespread congestion.

We compared behavior among a range of simulated network models, with various realism elements, ranging from very abstract (C0) to very realistic (C127). Our findings call into question the validity of previous studies [e.g., 5-14] that were based on abstract models that closely resemble C0. Those abstract models omitted TCP and variable router speeds, which are key elements responsible for shaping congestion spread in the modern Internet. Findings based on such abstract models provide little information about congestion behavior in the Internet.

7. Conclusions and Future Work

Over the past decade or so, many studies used simulation to investigate congestion spread in networks. Those studies often find that congestion can be modeled as a percolation process, spreading slowly under increasing load until a critical point. After the critical point, congestion spreads quickly throughout the entire network. Those same studies identified various measureable signals that arise around the critical point, which might allow one to predict onset of widespread congestion. These developments appear quite promising as a theoretical basis for monitoring regimes that network operators could deploy to warn of impending congestion collapse. Yet questions surround the studies, as the network models are quite abstract, bearing little resemblance to networks deployed based on modern technology. We explored these questions by examining the influence of realism on the spread of congestion in network simulations.

We began with an abstract network simulation, taken from the literature, and added elements of realism in various combinations, culminating with a high-fidelity simulation, also taken from the literature. From this study, we draw four main conclusions. First, we conclude that congestion spread in realistic network models differs significantly from spread in more abstract models. Even under heavy loads, realistic models limit the spread of congestion to the network edge, and retain connectivity in the network core. Further, realistic models deliver packets relatively successfully, and bound one-way packet latency to low values. None of these properties hold for abstract models used in the studies that we surveyed. Second, we conclude that models investigating network congestion must include TCP, along with hierarchically varied router speeds, before acceptable engineering findings can be established. In addition, where reasonable estimates of packet delivery and latency are required, packet dropping should be modeled with drop-tail FIFO queues. None of the studies we surveyed modeled TCP or hierarchically varied router speeds or drop-tail FIFO queues. Third, we conclude that modeling TCP is largely responsible for limiting congestion spread, for slowing decay of network connectivity, and for increasing probability of packet delivery. Further, hierarchically varied router speeds play a key role to ensure that the network core remains uncongested and intact, and packet dropping plays a key role to limit latency. Finally, we conclude that, using only graphs and packets, one can effectively visualize and compare global congestion behavior among a widely varied set of network models. We demonstrated an effective means to do so using: 2D plots, hierarchical clustering, and interactive multidimensional visualization.

Based on our findings, we conclude that the decade of studies we surveyed cannot be relied upon as guides to congestion spread in the Internet. We reach this conclusion because the studies contain models very similar to our abstract model. We doubt that the signals identified in those studies will actually appear on the Internet. We infer that the previously reported findings provide little information about congestion behavior in the Internet. We hope our study leads to better understanding of the influence of realism on congestion in network simulations, and to improved dialog throughout the diverse community of researchers who rely on network simulations.

We envision future work in three general directions. First, further research should explore our findings with respect to a collection of ISP-like topologies. While we believe our findings will hold, it appears prudent to verify that. Additionally, one could attempt

to expand the scope of our topologies to include multiple ISPs interconnected as a collection of autonomous systems. A second direction for further research is to consider whether random failures in the network core, coupled with alternate routing, could lead to cascading congestion that might consume the entire network. If such failure scenarios can be created plausibly, then one could determine if those scenarios might be modeled as a percolation process, spreading slowly under increasing (failure-induced) load until a critical point, after which the failure cascade spreads quickly throughout the entire network. Third, if such percolation processes can be identified, then one could seek precursor signals arising around the critical point. If such precursor signals exist, then they might serve as the basis for early warning of failure-induced congestion collapse. In this case, the theoretical findings from the earlier studies could be repurposed to solve a problem that might actually arise in a modern communications network. In addition, the general theory of percolation on a graph might provide a suitable basis to model macroscopic behavior in complex information systems other than the Internet.

8. Acknowledgments

We appreciate financial support and encouragement provided by our laboratory management. The technical staff benefit greatly from management's visionary thinking.

We thank researchers who, over the past decade or so, studied applicability of graph theory and percolation theory to macroscopic congestion in communication networks. Their studies provided an interesting intellectual glimpse into how monitoring regimes might raise early warning of impending congestion collapse. Their work inspired us to plan a program of research to explore the practicality of the ideas in real networks. Our current study embodies an initial plank in that program.

We also thank Phillip Gough of the Commonwealth Scientific and Industrial Research Organization. Phil's brilliant interactive multidimensional visualization enabled us to explore the details of congestion spread in our simulated topology. We were quite lucky that Phil's six-month stay as a visiting researcher at the National Institute of Standards and Technology overlapped with our work, and that our colleague Sandy Ressler put us in contact with Phil.

We also appreciate Phil's willingness to review our manuscript and provide suggestions for improvement. Similarly, we benefited from review and suggestions provided by colleague Guo Yang, who recently joined our research group from Bell Labs. And prior to conducting the study, we were fortunate to get advice from colleague Jim Filliben, who reviewed our experiment design.

References

[1] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez and D.-U. Hwang, Complex networks: structure and dynamics, in Physics Reports, Vol. 424, 2006, pp. 175-308.
http://dx.doi.org/10.1016/j.physrep.2005.10.009

[2] D. Stauffer and A. Aharony, Introduction to percolation theory: revised second edition, Taylor & Francis, 1994, 182 pages.
http://dx.doi.org/10.1016/S0378-4371(00)00536-7

[3] R. Albert, H. Jeong and A.-L. Barabasi, Error and attack tolerance of complex networks, in Nature, Vol. 406, 2000, pp. 378-382.
http://dx.doi.org/10.1038/35019019

[4] J. Doyle, D. Alderson, L. Li, S. Low, M. Rougan, S. Shalunov, R. Tanaka and W. Willinger, The "robust yet fragile" nature of the internet, in Proceedings of the National Academy of Sciences, Vol. 102, No. 41, 2005, pp. 14497-14502.
http://dx.doi.org/10.1073/pnas.0501426102

[5] R. Solé and S. Valverde, Information transfer and phase transitions in a model of internet traffic, in Physica A, Vol. 289, 2001, pp. 595-605.
http://dx.doi.org/10.1016/S0378-4371(00)00536-7

[6] M. Woolf, D. Arrowsmith, R. Mondragon and J. Pitts, Optimization and phase transitions in a chaotic model of data traffic, in Phys Rev E, Vol. 66, 2002, 046106.
http://dx.doi.org/10.1103/PhysRevE.66.046106

[7] D. Arrowsmith, R. Mondragon, J. Pitts and M. Woolf, Phase transitions in packet traffic on regular networks: a comparison of source types and topologies, Report 08, ISSN 1103-467X, Institut Mittag-Leffler, 2004, 8 pages.
https://www.mittag-leffler.se/preprints/files/IML-0405f-08.pdf

[8] G. Mukherjee and S. Manna, Phase transition in a directed traffic flow network, in Phys Rev E, Vol. 71, No. 6, 2005, 066108.
http://dx.doi.org/10.1103/PhysRevE.71.066108

[9] A. Lawniczak, P. Lio, S. Xie and J. Xu, Study of packet traffic fluctuations near phase transition point from free flow to congestion in data network model, in Proceedings of Canadian Conference on Electrical and Computer Engineering, 360-363, April 22-26, 2007, pp. 360-363.
http://dx.doi.org/10.1109/CCECE.2007.93

[10] B. Tadic, G. Rodgers and S. Thurner, Transport on complex networks: flow, jamming and optimization, in the International Journal of Bifurcation and Chaos, Vol. 17, No. 7, 2007, pp. 2363-2385.
http://dx.doi.org/10.1142/S0218127407018452

[11] D. Wang, N. Cai, Y. Jing and S. Zhang, Phase transition in complex networks, in Proceedings of the American Control Conference, June 10-12, 2009, pp. 3310-3313.
http://dx.doi.org/10.1109/ACC.2009.5159994

[12] Y. Rykalova, L. Levitan and R. Browe, Critical phenomena in discrete-time interconnection networks, in Physica A, Vol. 389, 2010, pp. 5259-5278.
http://dx.doi.org/10.1016/j.physa.2010.06.045

[13] S. Sarkar, K. Mukherjee, A. Ray, A. Srivastav and T. Wettergren, Statistical mechanics-inspired modeling of heterogeneous packet transmission in communication networks, in IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, Vol. 42, No. 4, August 2012, pp. 1083-1094.
http://dx.doi.org/10.1109/TSMCB.2012.2186611

[14] P. Echenique, J. Gomez-Gardenes and Y. Moreno, Dynamics of jamming transitions in complex networks, in Europhys Lett, Vol. 71, No. 2, 2005, 325.
http://dx.doi.org/10.1209/epl/i2005-10080-8

[15] K. Mills, E. Schwartz and J. Yuan, How to model a TCP/IP network using only 20 parameters, in Proceedings of the 2010 Winter Simulation Conference Dec. 5-8, 2010, pp. 849-860.
http://dx.doi.org/10.1109/WSC.2010.5679106

[16] K. Mills, J. Filliben, D. Cho, E. Schwartz and D. Genin, Study of proposed internet congestion control algorithms, NIST Special Publication 500-282, May 2010, 534 pages.
http://www.nist.gov/itl/antd/upload/NIST-SP-500-282.pdf

[17] E. Dijkstra, A note on two problems in connexion with graphs, Numerische mathematik, Vol. 1, No. 1, 1959, pp. 269-271.

[18] C. Dabrowski and K. Mills, FxNS Response Graphs and Cluster Analyses, October 2015, 147 pages.
http://www.nist.gov/itl/antd/upload/FxNSgraphs.pdf

[19] P. Gough, Interactive Multidimensional Visualization of FxNS Simulation Results, October 2015.
http://tinyurl.com/payglq6

[20] K. Mills and J. Filliben, Comparison of two dimension-reduction methods for network simulation models, Journal of Research of the National Institute of Standards and Technology, 116-5, September-October 2011, pp. 771-783.
http://dx.doi.org/10.6028/jres.116.020

Appendix A. Verification of FxNS Implementation of EGM and MesoNet

To verify that FxNS correctly implements EGM and the seven MesoNet realism elements, we compared results from FxNS against both EGM and MesoNet. We compared EGM results against FxNS results when all realism elements are disabled. We compared MesoNet results against FxNS results when all realism elements are enabled. Below, we document these comparisons.

A.1 Verification of EGM

We used FxNS, with all realism elements disabled, to repeat experiments of the Echenique team. Here we plot (Fig. A1) only two values of h (1 and 0.85) as p ranges from 1 to 30. Our plot shows the same behavior reported by EGM. When h = 1 congestion undergoes a second-order phase transition (starting at p = 2). When h = 0.85 congestion undergoes a first-order phase transition (starting at p = 9).

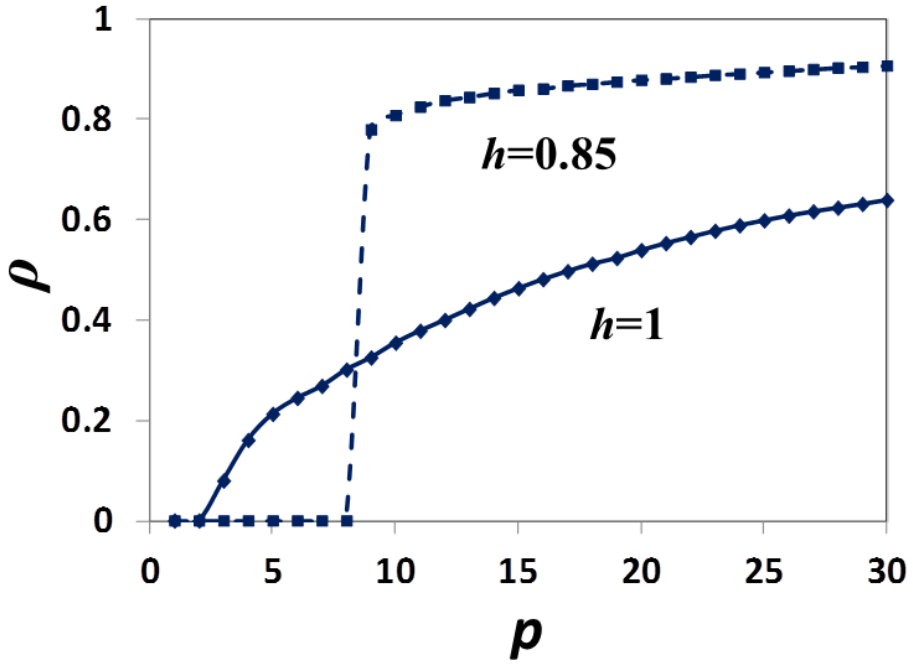

Figure A1. Results from our replication of simulations by Enchenique et al. [14]

Next, we investigated whether this phase-transition behavior also exists with a smaller topology, adapted from an ISP. Figure A2 shows the ISP topology, which consists of only 218 nodes. This is the same topology given in Fig. 3, but with node classification removed. We repeated our simulations using this topology.

We plot the outcome in Fig. A3. As with the AS topology, when h = 1 a second-order phase transition occurs and when h = 0.85 the phase transition is first-order. In the case of the smaller topology the onset of congestion begins around p = 2 regardless of the value of h. These results indicate that we can use this topology for our experiments without losing the main behavior of the EGM model.

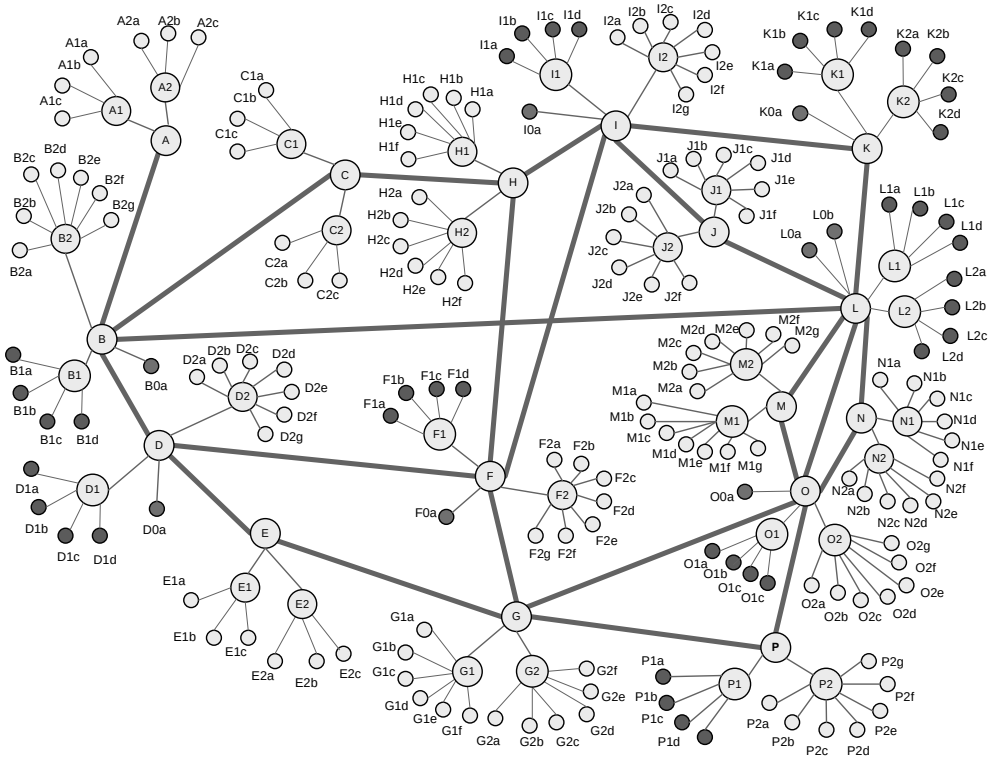

Figure A2. 218-node topology adapted from an Internet service provider

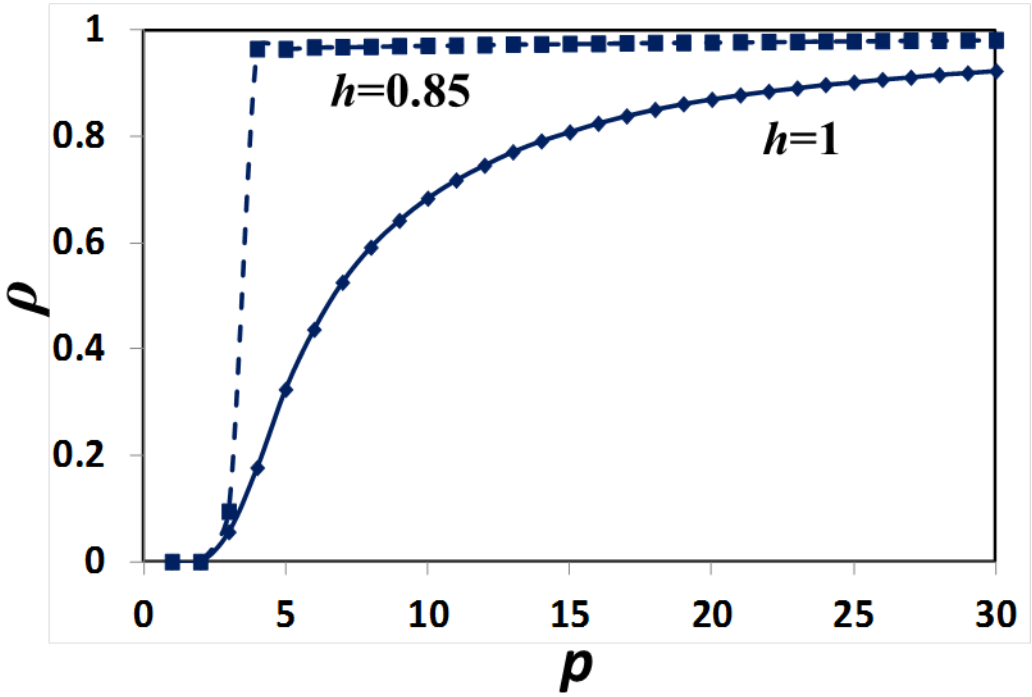

Figure A3. Results from FxNS simulations (no realism) within 218-node topology

A.2 Verification of MesoNet

We compared simulation results from MesoNet against FxNS with all realism elements enabled. We fixed FxNS parameters associated with realism elements to values identified in the enabled column of Table 5. We set MesoNet parameters to the same values. For both MesoNet and FxNS, we simulated packet-injection rates (p) from 1 to 5000. For each data point, we simulated 600 000 ts. We compare results with respect to seven orthogonal response dimensions that MesoNet exhibits [20]. The plots demonstrate that we correctly implemented MesoNet realism elements into FxNS.

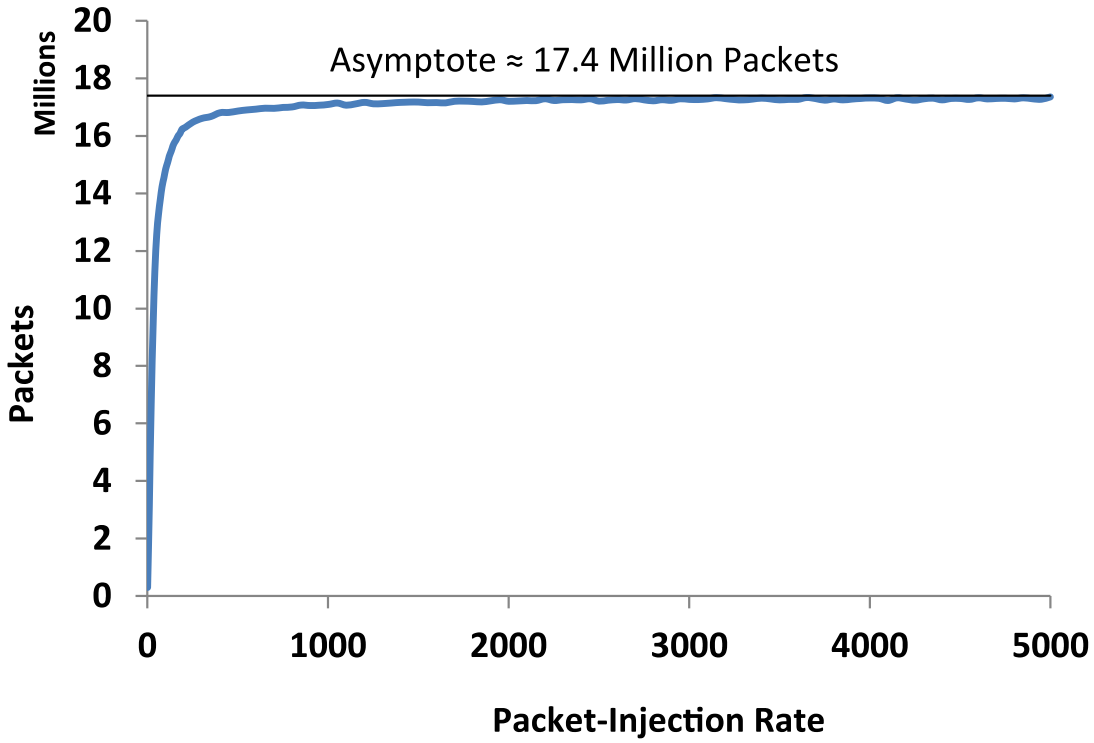

Figure A4. Aggregate packet throughput in the last 300 000 ts simulated by MesoNet

Figure A4 plots, for each injection rate, the aggregate number of packets delivered from the network (i.e., total throughput) in the last 300 000 ts of each simulation. These results are from MesoNet. Figure A5 shows the same results from FxNS. Note that both simulations show throughput increasing rapidly with p until reaching a maximum, just below 18 million packets. Thus FxNS mirrors the throughput behavior of MesoNet. Also of note: both models reach a maximum that does not increase with p. This occurs because the number of sources is fixed and once all sources are active, only so many packets can move through the network, and TCP adapts transmission rates to match perceived network capacity. This result indicates that when a network model is quite realistic, then maximum throughput is bounded and the network regulates itself to achieve that bound.

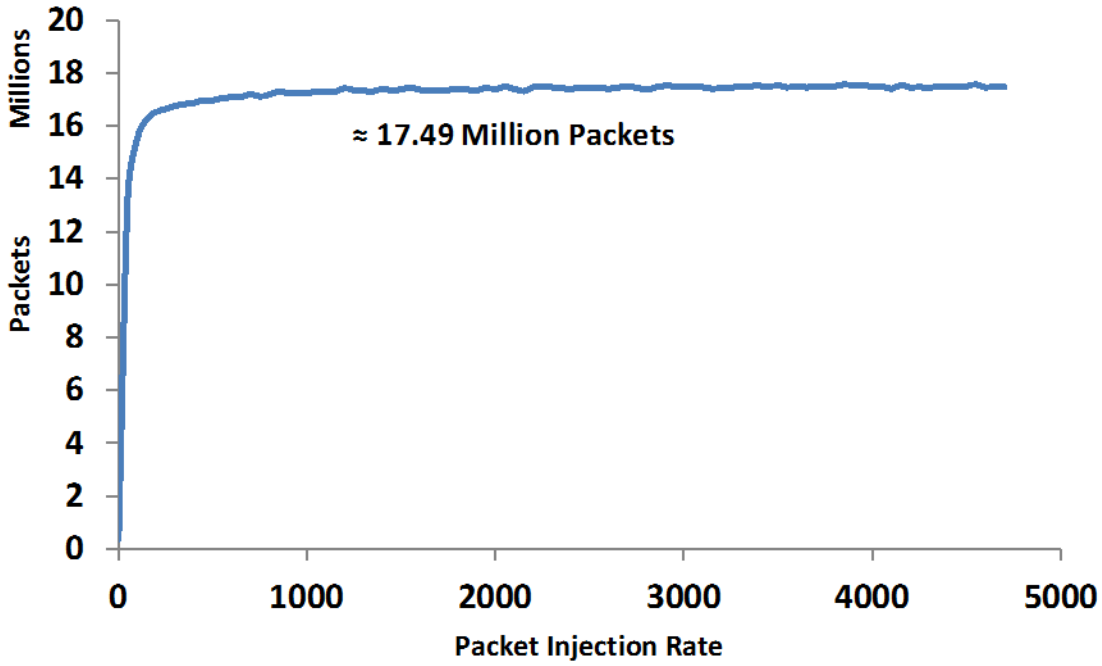

Figure A5. Aggregate packet throughput in the last 300 000 ts simulated by FxNS

Figure A6 plots, for each injection rate, the number of flows completed (i.e., aggregate flow throughput) by MesoNet in the last 300 000 ts of each simulation. Figure A7 gives the results for FxNS simulations. Both plots show flow completions increasing rapidly with *p* until reaching a maximum, which occurs at about 42 000 flows.

Figure A8 shows retransmission rate for TCP data segments for each injection rate simulated by MesoNet. A higher proportion or retransmissions denotes more trouble delivering packets, typically because packets or their acknowledgements are discarded or unduly delayed due to queue buildup. Figure A9 shows retransmission rate for FxNS simulations. Both plots show retransmission rate increasing rapidly, and perhaps heading to some maximum. While the rate of MesoNet is somewhat higher (around 42.5 %) than for FxNS (around 40 %), both curves have similar shape. The FxNS plot appears to be still rising, while the MesoNet curve appears to be leveling off. This suggests that the MesoNet queues can become somewhat more occupied than FxNS queues. Overall, though, retransmission rate appears reasonably similar between MesoNet and FxNS.

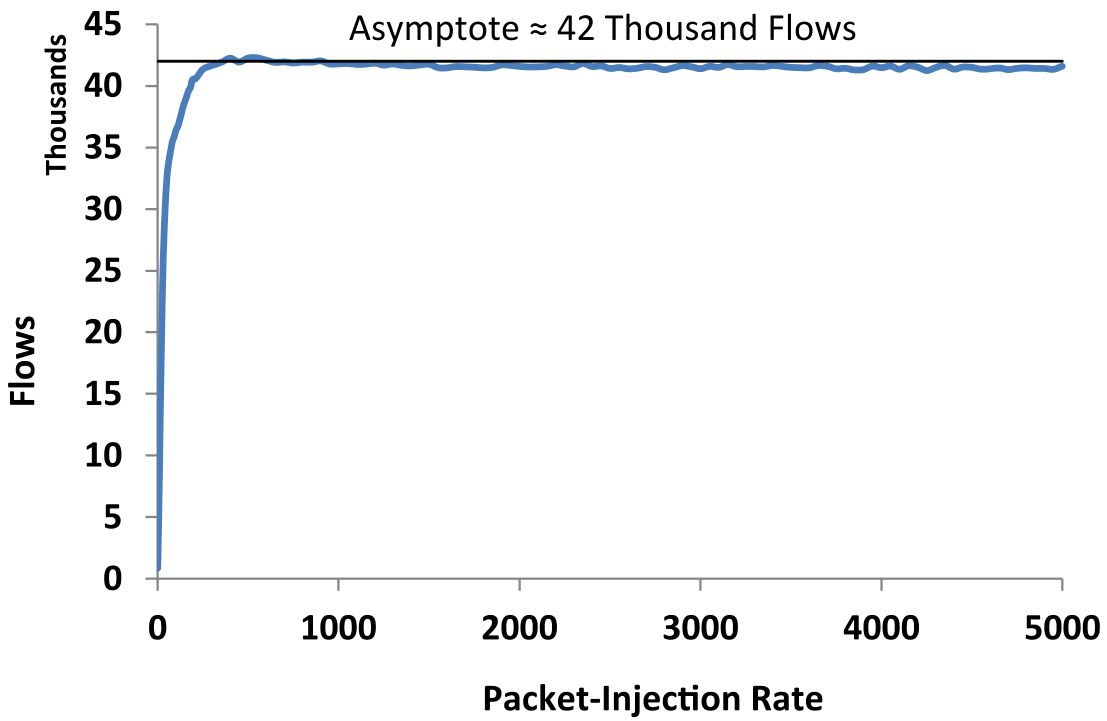

Figure A6. Aggregate flows completed in the last 300 000 ts simulated by MesoNet

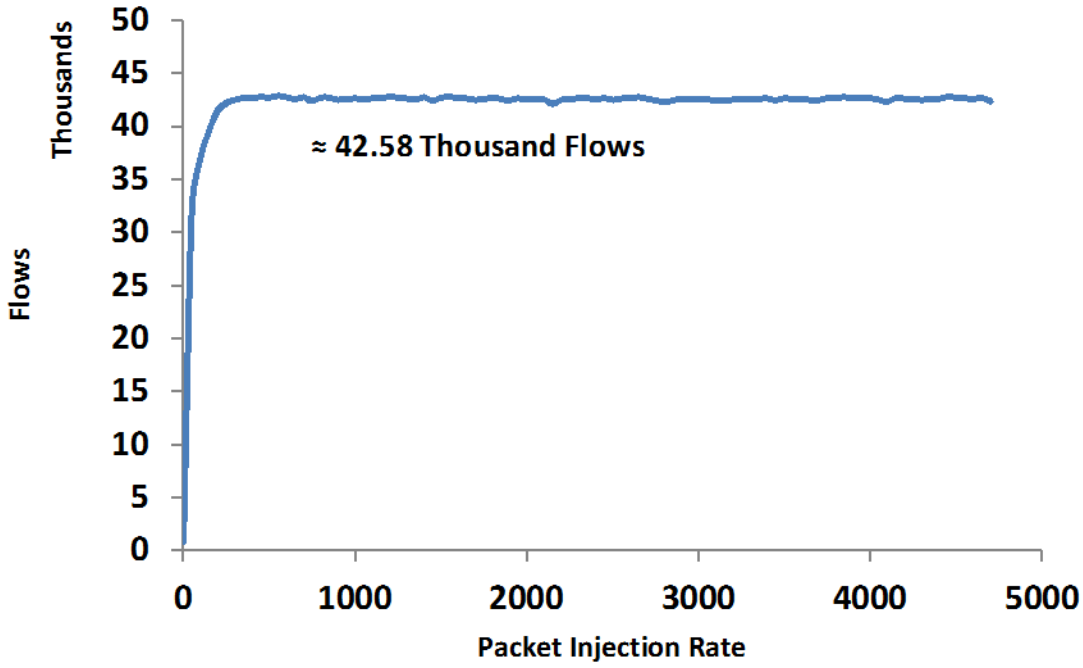

Figure A7. Aggregate flows completed in the last 300 000 ts simulated by FxNS

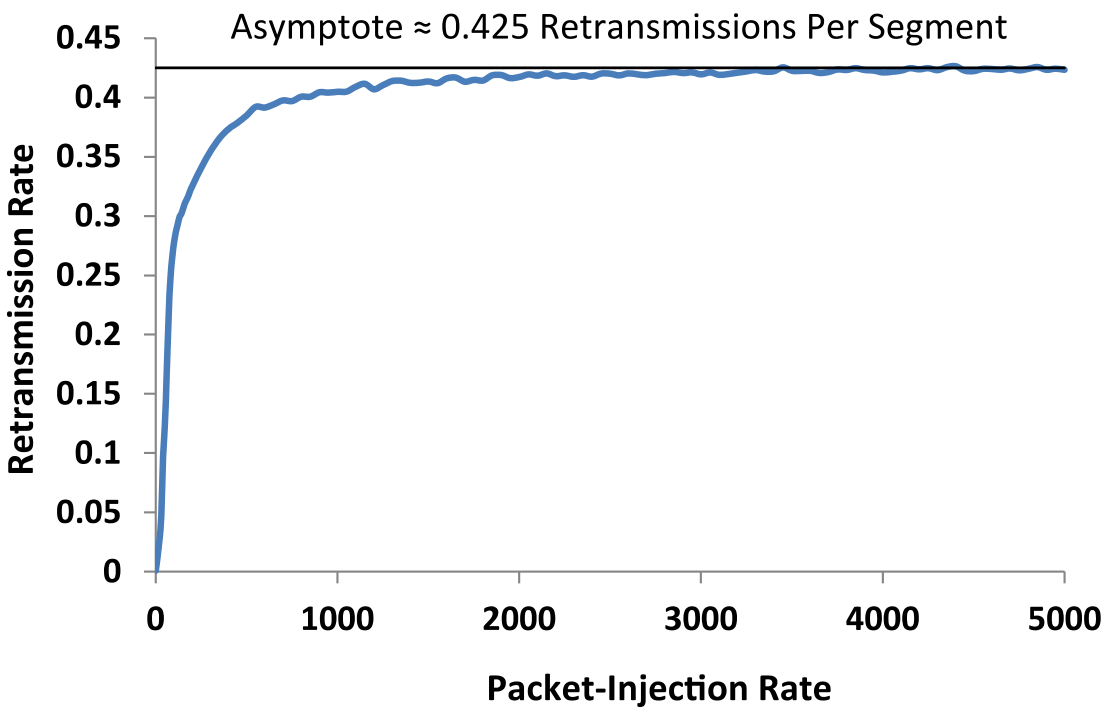

Figure A8. Retransmission rate for TCP data segments simulated by MesoNet

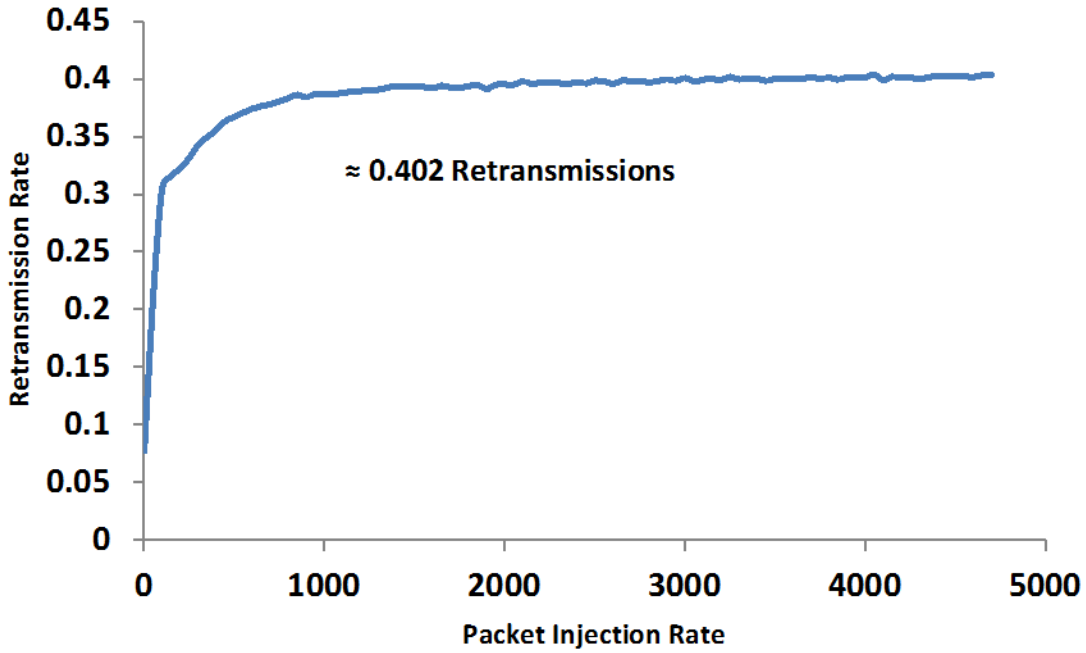

Figure A9. Retransmission rate for TCP data segments simulated by FxNS

Figure A10 plots average smoothed round-trip time (SRTT). SRTT is a measured estimate of average round-trip delay between sources and receivers. SRTT largely reflects queuing delays. Figure A11 shows the same plot for FxNS simulations. The shape of the curves agrees, and both approach a maximum. The SRTT plots suggest that MesoNet creates somewhat more congestion, as reflected by queue lengths, than FxNS.

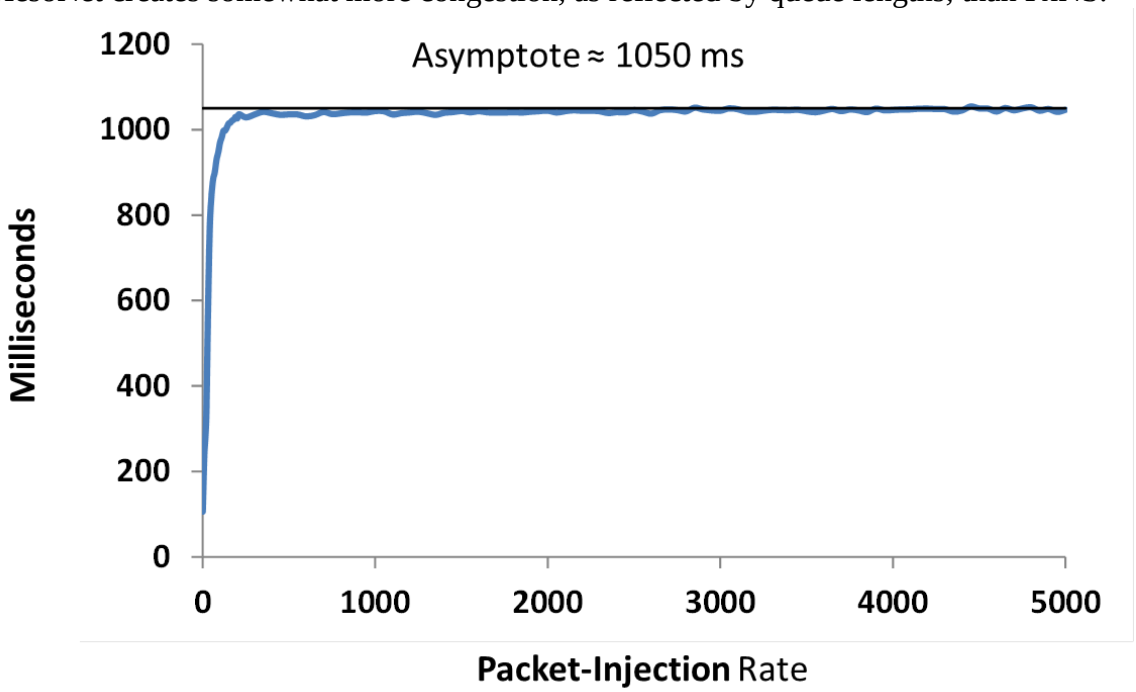

Figure A10. Average smoothed round-trip time simulated by MesoNet

Figure A12 reports, as simulated by MesoNet, average-per flow throughput for completed flows in three different classes. DD flows have highest throughput, as they transit speedy access routers that are directly connected to backbone routers. Such flows experience relatively little congestion, and so average throughput remains high, though somewhat variable, even as packet-injection rate increases to very high values. On the other hand, NN flows, which must transit the slowest access routers, see their average throughputs plummet quickly as *p* increases. The FF flows, which transit access routers that are a bit faster than normal, show a slower decline in average throughput as *p* increases. Further, once the network congests, FF flows achieve about seven p/ts, while NN flows achieve about 2/3 a p/ts. Figure A13 shows the same information plotted from FxNS simulations. The results are similar to the MesoNet results.

Though not included in essential MesoNet responses [20], we decided to also compare MesoNet and FxNS on the number of flows that could be completed in each of three classes (DD, FF, and NN). Figure A14 shows the results from MesoNet simulations and Fig. A15 plots the results from FxNS simulations. The shapes of the curves are similar for equivalent flow classes, and maximum rate of flow completions is quite close.

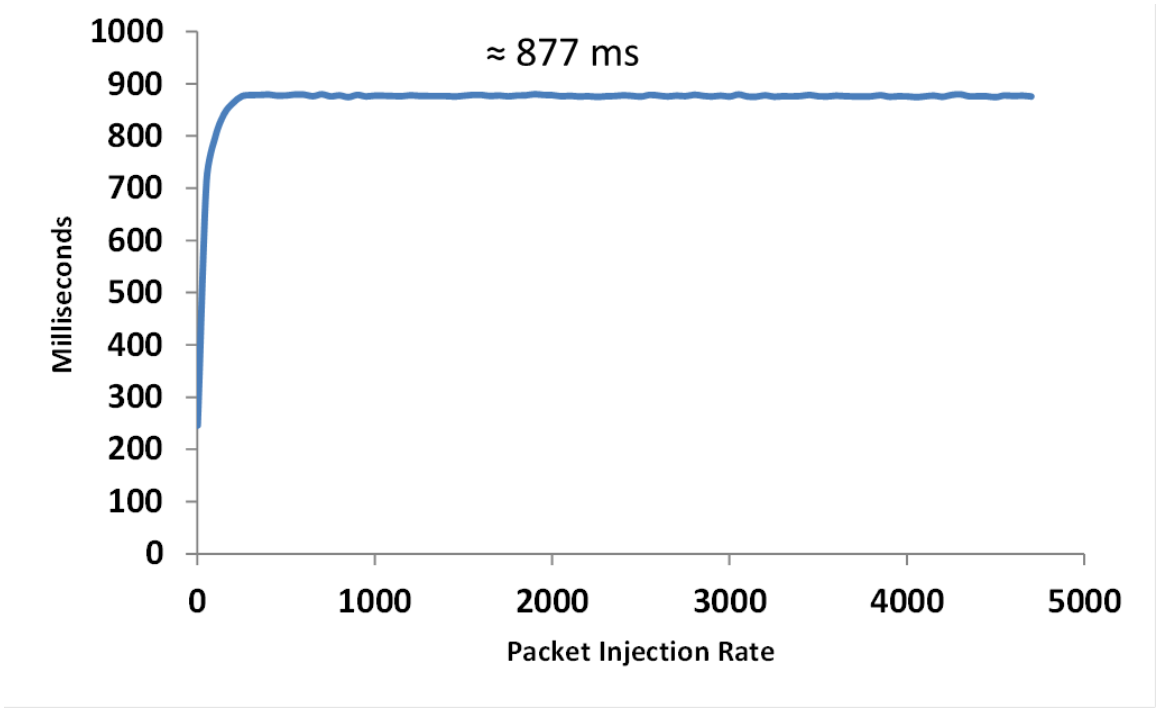

Figure A11. Average smoothed round-trip time simulated by FxNS

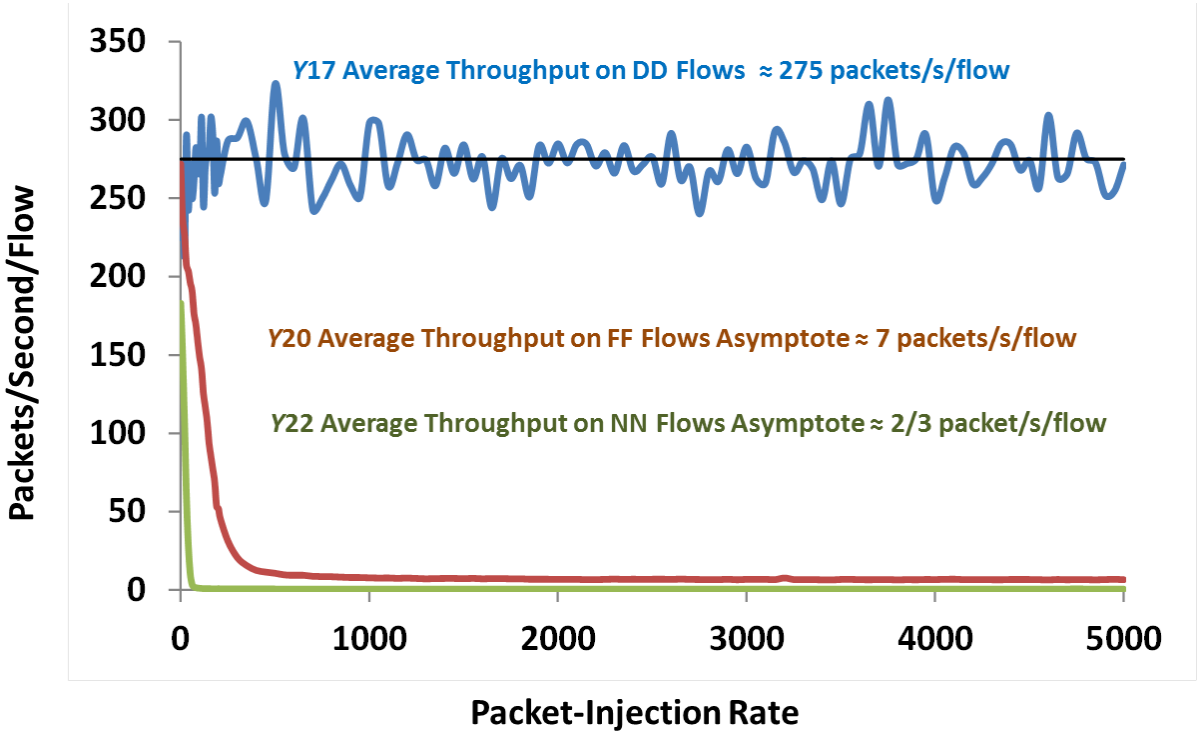

Figure A12. Average per-flow throughput for completed flows in three classes, as simulated by MesoNet

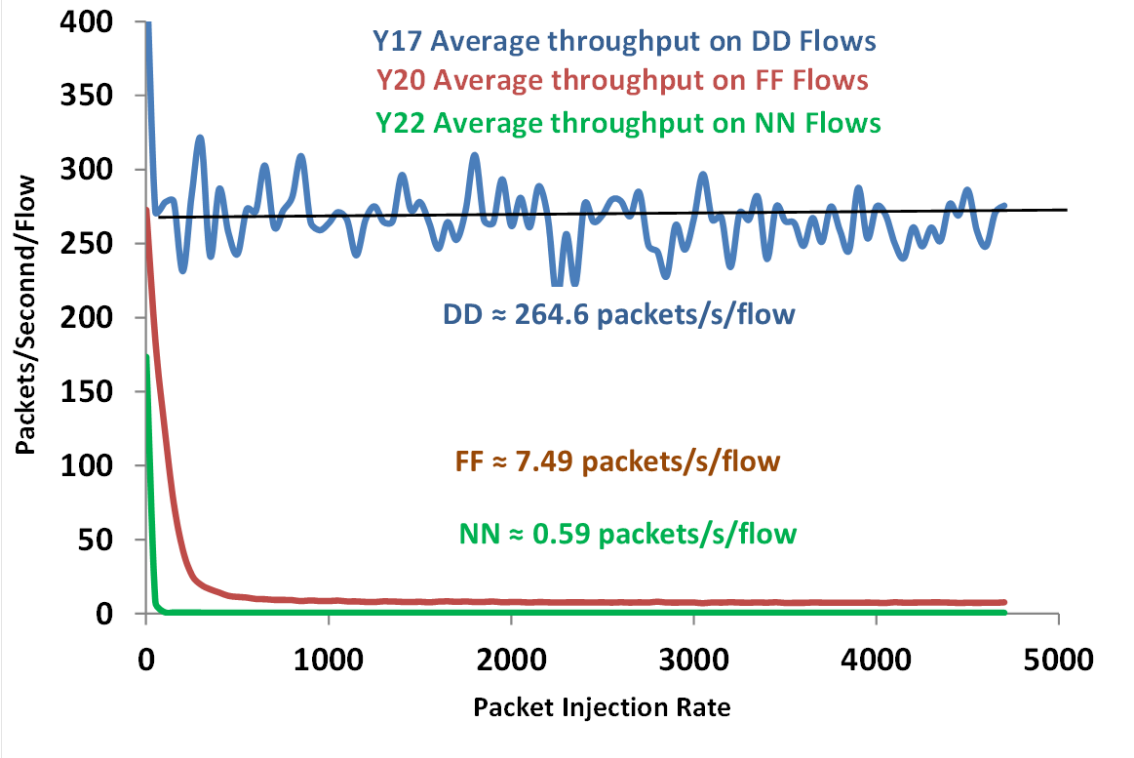

Figure A13. Average per-flow throughput for completed flows in three classes, as simulated by FxNS

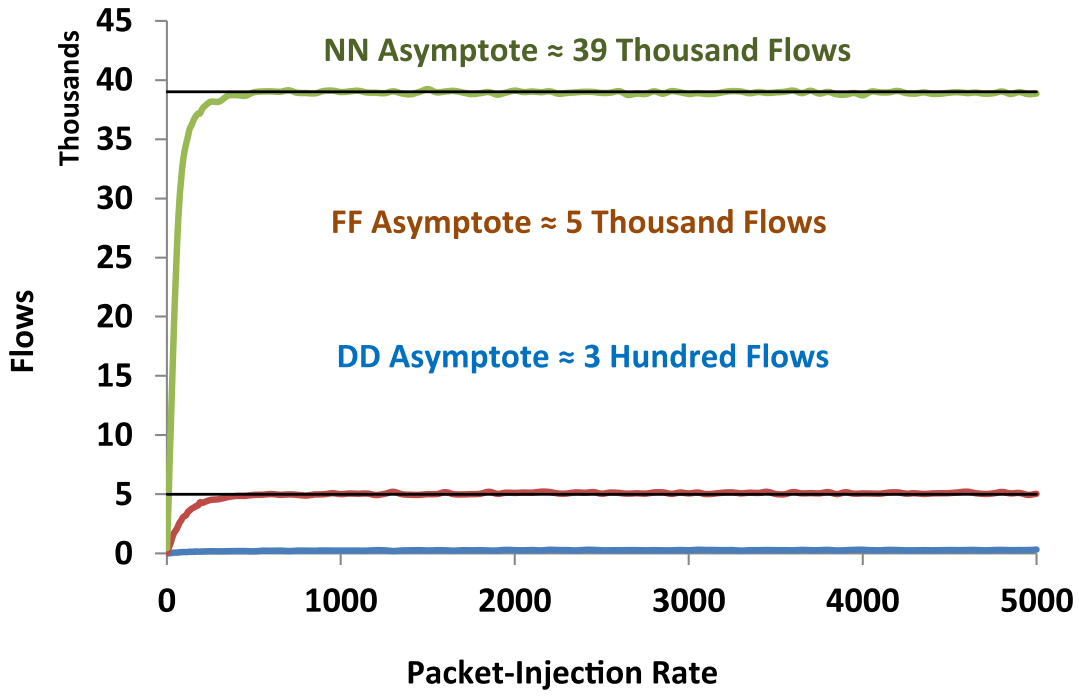

Figure A14. Completed flows in three classes, as simulated by MesoNet

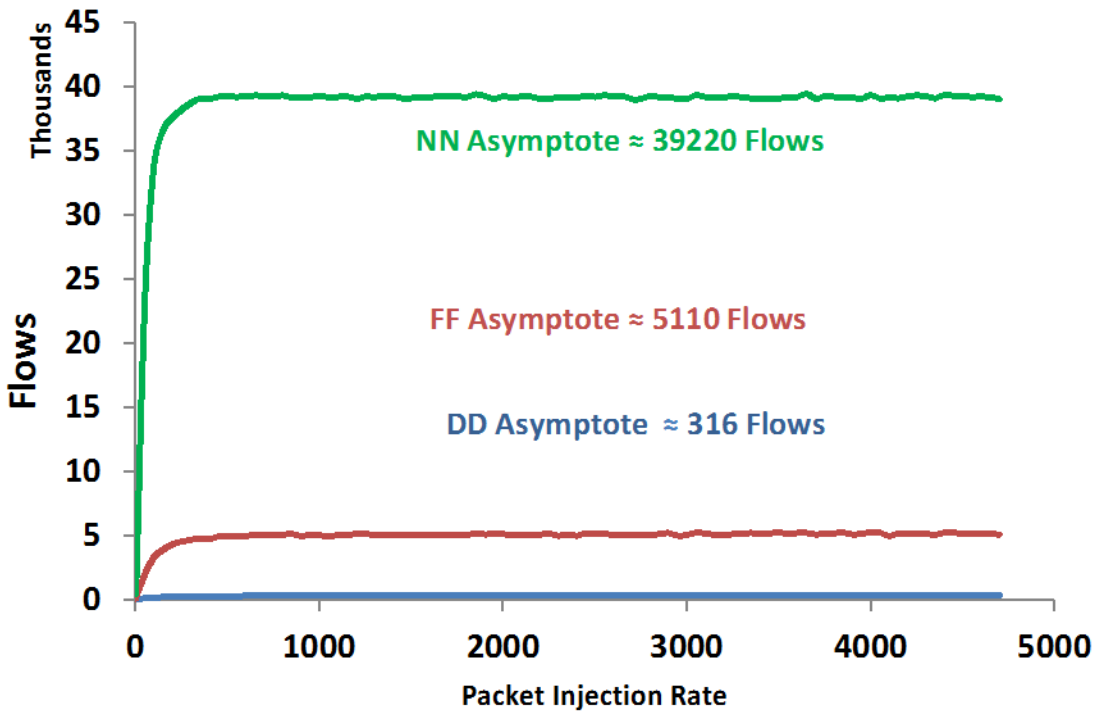

Figure A15. Completed flows in three classes, as simulated by FxNS

Overall, simulation results for MesoNet and FxNS are similar for the eight responses we compared. The shapes of plots for each response are aligned. For most responses, quantitative values from FxNS are quite close to those from MesoNet. MesoNet apparently creates a bit more congestion, which appears as larger packet queues. Due to this, MesoNet retransmission rates and SRTTs are somewhat higher at high packet-injection rates. The comparison of simulation results leads us to conclude that FxNS correctly implements MesoNet realism factors.

Appendix B. LSS Congested Nodes

Recall that Fig. 6 showed 34 plots of χ (y axis) vs. p (x axis), one for each FxNS configuration. Here, χ is the proportion of nodes in the LSS of isolated nodes, i.e., nodes that were both congested and cutoff. Most of the studies we surveyed focused only on congested nodes. We could also have measured congestion spread using only the LSS of congested nodes, i.e., ignoring cutoff nodes.

Figure B1 shows 34 plots of γ (y axis) vs. p (x axis), one for each FxNS configuration. Here, γ is the proportion of nodes in the LSS of congested nodes. Comparing plots in Fig. B1 against plots in Fig. 6 illustrates that for configurations that percolate (i.e., where χ and γ reach 1), the LSS of isolated nodes spreads more quickly at low injection rates (p) than does the LSS of congested nodes. In most cases, percolation for the LSS of congested nodes happens suddenly at higher values of p. Despite these differences, the same underlying factors determine whether or not congestion spreads widely. We discuss this next.

Figure B2 clusters vectors from Fig. B1. Comparing Figure B2 against Fig. 12, which shows clustering of the LSS of isolated nodes, confirms the same underlying factors. Figure B1 and Fig. 12 both show large distances between two main clusters: one where configurations enabled TCP or VS or both and another where configurations disabled both TCP and VS. This evidence supports our findings that both variable router speeds and TCP influence congestion spread. Variable router speeds influence congestion spread by limiting congestion to the network edge. TCP influences congestion spread by detecting congestion and reducing packet-injection rate accordingly. TCP and variable router speeds are critical to model in any simulation that intends to produce congestion patterns consistent with those in communication networks based on modern Internet technology.

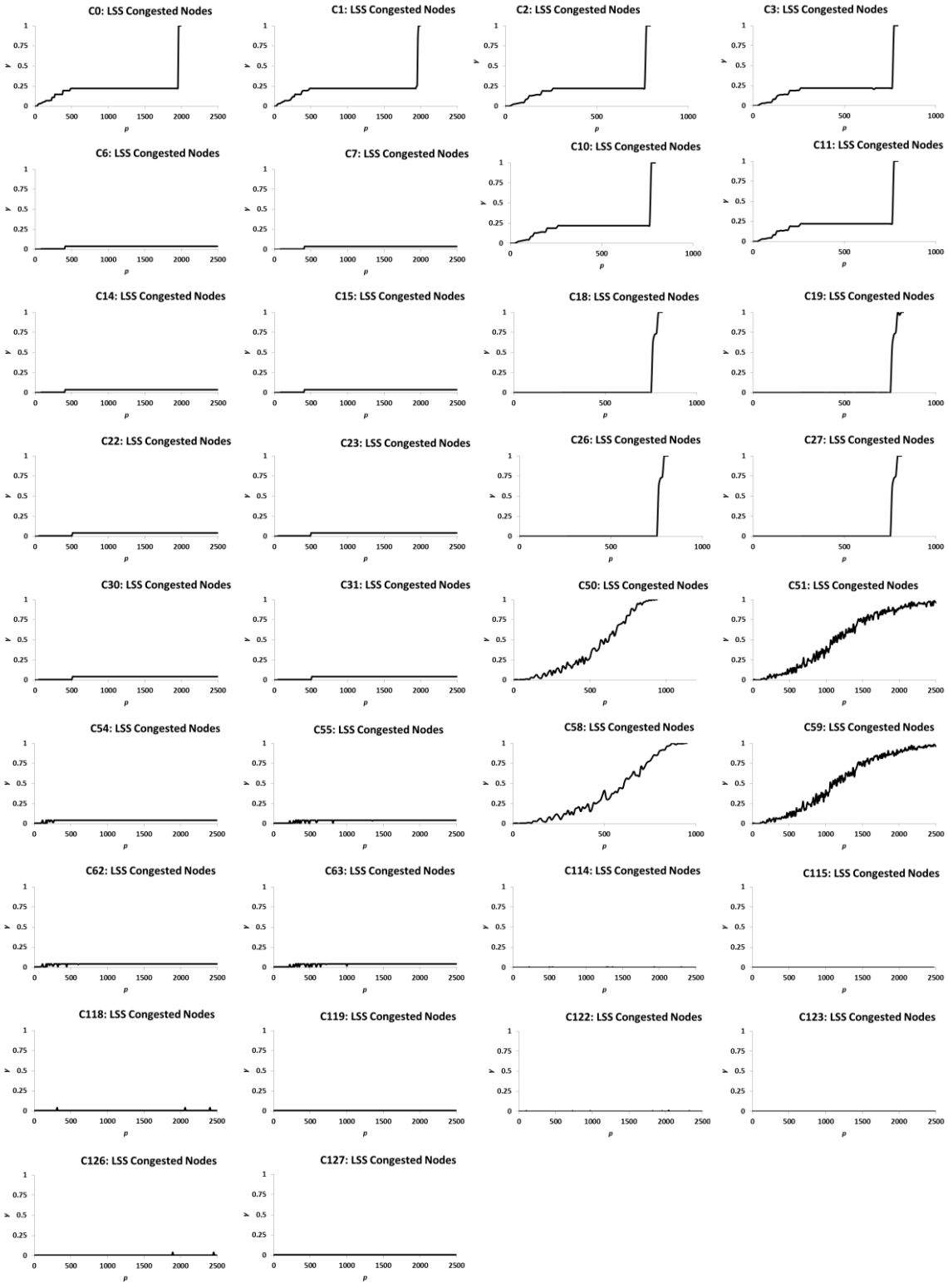

Figure B1. Proportion of nodes in LSS of congested nodes (γ) for 3 FxNS configurations

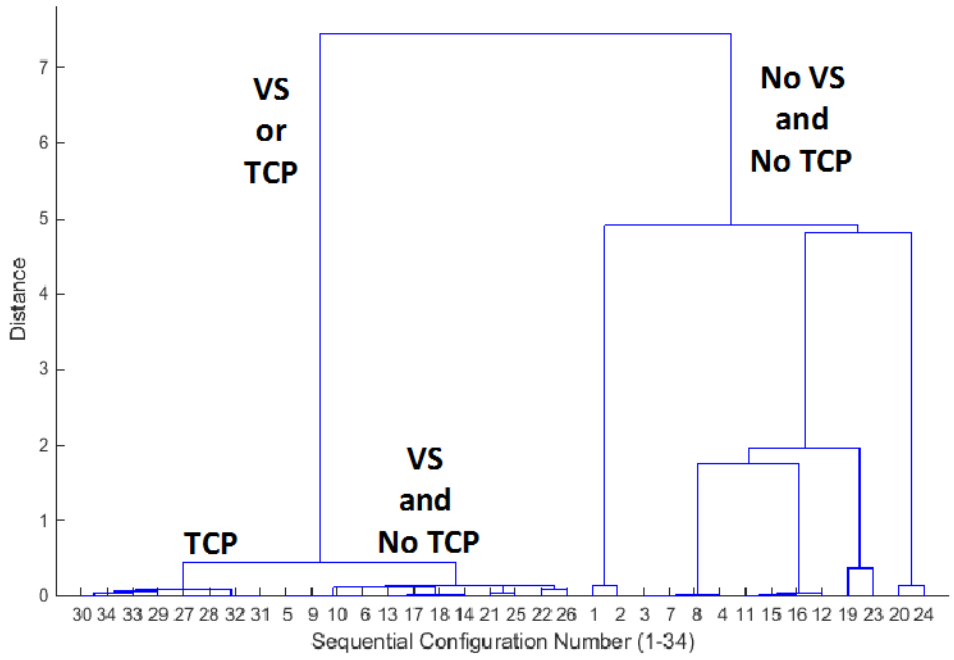

Figure B2. Clustering of LSS congested nodes (γ) based on Squared Euclidean Distance