The Need for Realism when Simulating Network Congestion

Kevin Mills
NIST
Gaithersburg, MD 20899
kmills@nist.gov

Chris Dabrowski
NIST
Gaithersburg, MD 20899
cdabrowski@nist.gov

ABSTRACT
Many researchers use abstract models to simulate network congestion, finding patterns that might foreshadow onset of congestion collapse. We investigate whether such abstract models yield congestion behaviors sufficiently similar to more realistic models. Beginning with an abstract model, we add elements of realism in various combinations, culminating with a high-fidelity simulation. By comparing congestion patterns among combinations, we illustrate congestion spread in abstract models differs from that in realistic models. We identify critical elements of realism needed when simulating congestion. We demonstrate a means to compare congestion patterns among simulations covering diverse configurations. We hope our contributions lead to better understanding of the need for realism when simulating network congestion.

Author Keywords
Congestion; criticality; networks; percolation; simulation

ACM Classification Keywords
I.6.1 SIMULATION AND MODELING: Model Validation and Analysis

1. INTRODUCTION

The science of complex networks [1] has matured to the point where one can study mathematical structure for many classes of probabilistic graphs (e.g., random, scale-free, small-world), as well as dynamical processes [2] moving within such graphs. Typically, abstractions are adopted in order to model real networks using techniques (e.g., graph theory and percolation theory) available from network science. Tension arises when such powerful abstractions are used to study real networks. How can one be sure that chosen abstractions adequately embody key properties of a network under study? This question of model validation motivates the work reported here.

Many researchers [e.g., 3-12] use simulation to investigate congestion spread in network topologies, often finding congestion can be modeled as a percolation process on a graph, spreading slowly under increasing load until a critical point, after which congestion spreads quickly throughout the network. The researchers identify various signals that arise around the critical point. Such signals could foreshadow onset of widespread congestion. These developments appear promising as a theoretical basis for monitoring methods that could be deployed to warn of impending congestion collapse. Despite showing promise, questions surround this research, as the models are quite abstract, bearing little resemblance to communication networks deployed based on modern technology. We explore these questions by examining the influence of realism on congestion spread in network simulations.

We begin with an abstract network simulation from the literature. We add realism elements in combinations, culminating with a high-fidelity simulation, also from the literature. By comparing patterns of congestion among the combinations, we explore a number of questions. Does spreading congestion in abstract network models mirror spreading congestion in realistic models? How do specific elements of realism influence congestion spread? What elements of realism are essential to capture in models of network congestion? What elements are unnecessary? What measures of congestion can be compared, and how, across diverse network models?

We make three main contributions. First, we illustrate congestion spread in abstract models differs significantly from spread in realistic models. Second, we identify elements of realism needed when simulating congestion. Finally, we demonstrate a method to compare congestion patterns among diverse network simulations.

The remainder of the paper is organized in five sections. Section 2 reviews some related work where researchers use abstract models to investigate congestion spread in network simulations. Section 3 describes the configurable network simulator used in our experiment. The simulator can be configured to mirror an abstract model [12], a realistic model [13], and various intermediate combinations. Section 4 details our experiment design. We present and discuss results in Sec. 5. We conclude in Sec. 6.

2. RELATED WORK

Reviewing a decade of congestion studies [3-12] reveals many similarities, and some variations, among the abstract models used. Below we summarize the models along four dimensions: topology, traffic sources/sinks, routers and

SpringSim-CNS 2016 April 3-6 Pasadena, CA, USA
© 2016 Society for Modeling & Simulation International (SCS)

congestion measures. Elsewhere [14] we provide more details about each of the studies.

Researchers used either deterministic or probabilistic topologies. The most popular deterministic topology was a square lattice, either open [6, 11] or folded into a toroid [3-5, 7, 10]. Rykalova et al. [10] also used a ring. Echenique et al. [12] used a real topology taken from the Internet autonomous system map, circa 2001. Arrowsmith et al. [5] started with a 2D lattice and then generated triangular and hexagonal depleted lattices by probabilistically removing links. Other researchers used random processes to generate topologies: Erdős–Rényi [9], exponential [8], scale-free [8-9], or small world [9].

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 router [7-10, 12]. Sarkar et al. [11] restricted sources and sinks to the network edge, while Mukherjee and Manna [6] placed sources at the top edge of a lattice and sinks at the bottom edge. Other researchers [3-5] 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 [3-7, 10-11] was for each source to generate a packet per time step (p/ts) with a specified probability. A few studies [8-9, 12] generated a fixed number of packets/ts and randomly assigned the packets to sources. One study [8] 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 [3-7, 9-10, 12] queue discipline was unbounded first-in, first-out (FIFO) queues. One study [8] used bounded last-in, first-out (LIFO) queues. One study [11] used bounded FIFO queues, where the oldest packet was dropped when a packet arrived at a full queue. Most studies [3-6, 10, 11] selected next hop based on shortest-path first (SPF) in hops. Ties were broken either by shortest queue length [3-4, 11], link use [5] or tossing a fair coin [6, 10]. One study [7] selected next hop with the choice among three different SPF metrics: hops, queue length, or their sum. Two studies [9, 12] used SPF based on a weighted sum of hops and queue length. One study [8] used guided random walk to select next hops. In most studies [3-5, 8, 11-12] each router forwards one p/ts. In two studies [7, 10] each router forwards one p/ts for each queue. One study [6] has each router forward a batch of packets at each time step. One study [9] 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 [3-4, 6, 8]; packets delivered (i.e., aggregate throughput) [3-5]; queue lengths [4-6, 8]; packets in the network [7, 9-10, 12]; and packet drop rate [11]. Various studies analyzed the measures as time series, proportions, or variances.

Beyond the differences we identified above, the studies we surveyed shared many similarities. An abstract model is developed and then used to explore congestion in various topologies. Congestion spread is examined through selected measures. A critical load is identified, after which trajectory changes distinctly for selected measures. When examined by engineers, who deploy and manage networks based on Internet technology, the degree of abstraction is sufficiently high to call into question the findings. The topologies are rarely congruent with real Internet topologies [15], various parameter values are not consistent with real engineering choices, congestion-control protocols are not modeled and the distribution of packet injection is unlike patterns that occur with real users. Does this lack of realism matter? If so, what realism elements must be present to draw valid conclusions about congestion spread? We investigate these questions here.

3. MODELS

We conducted an experiment (see Sec. 4) with a simulation model we named FxNS (Flexible Network Simulator). FxNS is based on an abstract model, EGM, developed by Echenique, Gomez-Gardenes and Moreno [12]. We added a set of seven realism elements, factored from MesoNet [13]. While many realistic network simulators exist [16], we chose MesoNet because the model is terse (requiring only 20 parameters) and factors easily, and because the model scales (simulating up to ½ million nodes engaged in over 125×10^3 simultaneous flows).

We implemented the realism elements as options within FxNS. Since each element can be enabled or disabled, FxNS could support (2^7 =) 128 combinations. However, as explained in Sec. 3.3, we respect some dependencies among realism elements. As a result, FxNS supports only 34 combinations. FxNS can be configured to behave as EGM (most abstract model), as MesoNet (most realistic model), and any of the remaining 32 valid combinations intermediate between EGM and MesoNet. With all realism elements enabled, we use FxNS to simulate ¼ million nodes engaged in over 50×10^3 simultaneous flows. FxNS should scale up further, to the same order as MesoNet.

In Sec. 3.1 we describe EGM, and give simulation results demonstrating that FxNS correctly implements EGM. In Sec. 3.2 we describe MesoNet, and its 20 parameters spread among five categories. We also define our mapping from MesoNet parameters to FxNS realism elements. In Sec. 3.3, we justify dependencies adopted among realism elements and we describe our numbering convention for the FxNS

combinations used in our experiment. Elsewhere [14] we provide additional details on these topics.

3.1. Abstract Model

In EGM, p packets are injected at each time step (ts) with source and destination nodes for each packet chosen randomly (uniform). Injected packets are placed at the end of a source's unbounded FIFO packet queue. After injection, each node can forward one packet from its queue to a next node. If the next node is the destination, the packet is delivered; otherwise the next node is chosen as the neighboring node i with minimum δ_i as defined in eq. 1:

$$\delta_i = hd_i + (1-h)c_i \quad (1)$$

where i is index of a node's neighbor, d_i is minimum hops to the packet's destination via i, and c_i is queue length of i. When $h = 1$ the routing amounts to SPF hops. When $h < 1$, routing is congestion aware, as packets may follow routes longer in hops, but shorter in total queuing delay. The lower h the more congestion-aware routing becomes.

EGM measures congestion as ρ, the ratio of packet outflow to inflow as defined in eq. 2:

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

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

Using EGM with an 11 174-node topology, Echenique et al. [12] explored effects of SPF hops routing vs. congestion-aware routing as p increases. They found that for routing via SPF hops ρ undergoes a 2nd order transition as p passes a critical load, while under various degrees of congestion-aware routing ρ undergoes a 1st order transition as p passes critical load. Using our FxNS implementation of EGM, we replicated these results, as shown in Fig. 1.

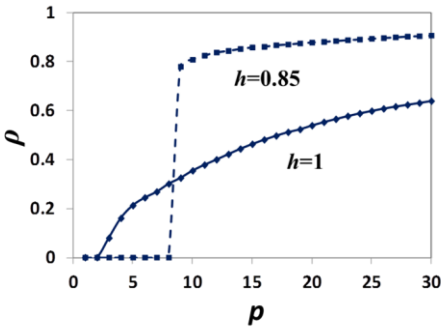

Figure 1. FxNS replication of EGM simulation results

3.2. Realistic Model

MesoNet provides a realistic TCP (Transmission Control Protocol) network model, requiring only 20 parameters spread across five categories, as shown in Table 1. Mills et al. [16] used MesoNet to compare congestion-control algorithms proposed for the Internet.

Category	ID	Name	FxNS
Network	x1	topology	NC
	x2	propagation delay	DE
	x3	network speed	VS
	x4	buffer provisioning	PD
Sources & Sinks	x5	number sources/sinks	SR
	x6	source distribution	
	x7	sink distribution	
	x8	source/sink speed	VS
Users	x9	think time	*p*
	x10	patience	n/a
	x11	web object file sizes	FL
	x12	larger file sizes	n/a
	x13	localized congestion	
	x14	long-lived flows	
Congestion Control	x15	control algorithm	TCP
	x16	initial *cwnd*	
	x17	Initial *sst*	
Simulation Control	x18	measurement interval	fixed
	x19	simulation duration	fixed
	x20	startup pattern	*p*

Table 1. MesoNet Parameters with Mapping to FxNS Elements

MesoNet allows for three-tier topologies of routers: core, point-of-presence (PoP), and access. In our experiment, we use an Internet service provider (ISP) topology shown in Fig. 2, which provides three types of access routers: D-class (red), F-class (green) and N-class. MesoNet defines speed relationships among all routers. Changing one parameter can scale network speed and higher router tiers can support the maximum input traffic expected from lower tiers. Sources and sinks can be placed below access routers as a fourth tier with ¼ million nodes (not shown in Fig. 2).

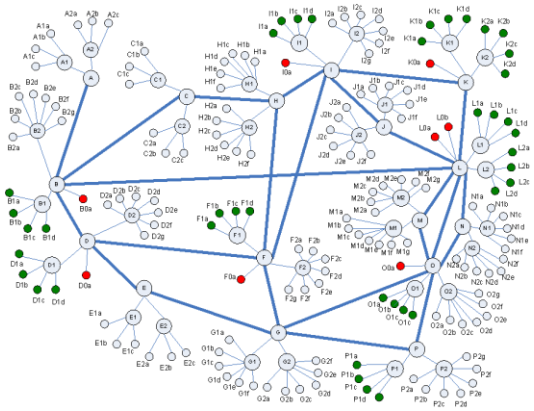

Figure 2. Three-tier 218-router topology – 16 core (A-P), 32 PoP (A1-P2) and 170 access (A1a-P2g)

FxNS maps router typing to realism element **NC** (*node classes*), which ensures that sources and sinks are placed only at the network edge. FxNS maps router speed scaling to

realism element **VS** (*variable speeds*). MesoNet allows sources and sinks to connect to the network at two different speeds: fast and normal. FxNS also maps these interface speeds to realism element VS. In MesoNet links between core routers have intrinsic propagation delays matched to geographic placement and physics. FxNS maps these to realism element **DE** (*propagation delays*). These intrinsic propagation delays were used to compute SPF routes for the network core. MesoNet also includes various buffer provisioning algorithms. FxNS uses only one (estimated round-trip time multiplied by router forwarding speed) and maps this to realism element **PD** (*packet dropping*).

MesoNet allows the number of sources and sinks to be scaled and also allows probabilistic placement of sources and sinks under various types of access router. MesoNet ensures there are four times as many sinks as sources. FxNS adopts these procedures and maps them to realism element **SR** (*sources and receivers*).

MesoNet provides a rich array of user parameters, but FxNS maps only two. First, MesoNet users have think time between initiating data transfers. FxNS replaces think time with packet-injection rate, *p*. Second, MesoNet allows users to randomly select the file size for each data transfer. FxNS maps this parameter to the **FL** (*flows*) realism element, which creates sets of packets transferred in a related stream. MesoNet allows users to exhibit limited patience when waiting for data transfers to complete, but in FxNS all users have infinite patience. MesoNet allows probabilistic selection of various larger file sizes and spatiotemporal congestion. FxNS does not implement these features.

MesoNet allows probabilistic assignment of congestion-control algorithm to individual sources/sinks. In FxNS only **TCP** (*transmission control protocol*) is used. MesoNet also allows specification of initial *cwnd* (congestion window) and *sst* (slow-start threshold). FxNS maps these parameters to realism element TCP.

Finally, MesoNet offers a set of three simulation control parameters. FxNS uses measurement interval size and duration (in measurement intervals) to bound simulation length. MesoNet also allows individual traffic sources to start in a specified pattern. FxNS subsumes this under packet-injection rate.

To verify FxNS correctly implements MesoNet realism elements, we conducted comparative simulations, running MesoNet and FxNS (with all realism elements enabled) for 600 000 ts using identical parameter values. As shown elsewhere [14], we compared model output for seven essential MesoNet responses [17].

3.3. Combination Models

While FxNS can enable and disable the seven realism elements shown in Table 1, some dependencies exist, as shown in Fig. 3. Starting with all realism elements disabled (EGM), one can easily enable packet dropping (PD) and node classes (NC). Variable speeds (VS) require routers to be classified by type. Similarly, propagation delays (DE) appear on core network links, which can be identified only through router types. While sources/sinks (SR) might be included as a second tier under a flat topology, i.e., without node classes, we decided to restrict them to a fourth tier under access routers. We took this decision for convenience, allowing us to eliminate 24 combinations that would otherwise need to be simulated. We imagined influence of sources/sinks could be discerned even with this restriction. Enabling flows (FL) means packets are injected as a stream between source and sink, thus FL requires SR. Finally, TCP regulates packet-transmission rate only on flows.

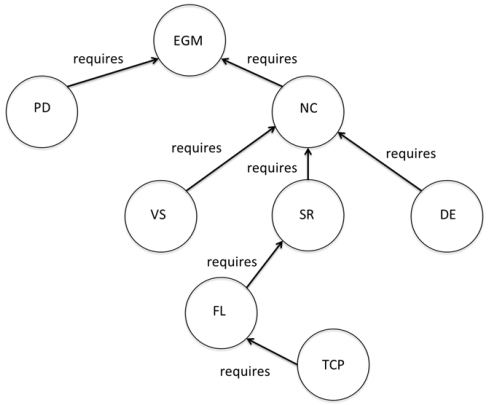

Figure 3. Dependencies among FxNS realism elements

Seq	Cmb	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
. . .								
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

Table 2. Elided list of valid FxNS combinations

We identify FxNS combinations by number, based on binary encoding, as shown in Table 2. Each realism element is assigned a position in a seven-bit vector, from most (bit 7 - TCP) to least (bit 1 - PD) significant. 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: the combination (Cmb) number. The most abstract combination is c0 and the most realistic is c127. Each combination is also assigned a sequence (Seq) number (1-34). Both numbers are used in discussing results.

4. EXPERIMENT DESIGN

We designed an experiment to explore influence of realism on congestion spread in a network simulated with FxNS. We identify fixed input parameters used in all simulations. We define parameters we vary. We define four responses measured for all simulations.

4.1. Fixed Input Parameters

We used the same 218-router topology (recall Fig. 2) in all simulations. We used Dijkstra's SPF algorithm to compute next hops for core routers based on propagation delays. Routing to/from core nodes consists of single paths with obvious next hops. Note that propagation delays are used to compute SPF next hops in the core regardless of whether DE is enabled or disabled.

We execute each simulation for a target of 200 000 ts. Individual simulations can self-adapt to execute fewer ts in order to limit memory usage when PD is disabled. No simulation executed fewer than 41 400 ts.

4.2. Variable Input Parameters

We varied only two parameters: (1) combination and (2) packet-injection rate *p*. For each combination, FxNS simulates a set of enabled/disabled realism elements (recall Table 2). Table 3 gives parameter values assigned to each element when enabled and disabled.

For each combination simulated, we varied *p* up to 2500. When extreme congestion appears at successive values of *p*, simulation of a combination could self-terminate. This saves computation time because once a combination demonstrates extreme congestion for several increasing values of *p* then the combination will continue to exhibit congestion as *p* increases. In no case did a simulation terminate a combination before *p* passed 790.

4.3. Responses

We chose responses that could be usefully compared across all simulated combinations: most abstract to realistic. We determined that all combinations shared two measurable concepts: graphs and packets. Using these we measure: congestion spread (χ), network connectivity (α) and effectiveness (π) and efficiency (δ) of packet delivery. All responses fall in the interval [0...1]. We measure each response for each combination at each packet-injection rate. We define these responses precisely elsewhere [14]. Here we give intuitive definitions.

Each of our simulated topologies is a graph of nodes connected by links, where the entire graph G_N contains $|G_N|$ nodes. We label a node congested whenever queued packets exceed 70 % of 250×router forwarding speed. When fewer packets are queued, we label a node uncongested. We label any uncongested node as cutoff when it links only to congested neighbors. After labeling, we compute connected subgraphs of nodes that are either congested or cutoff. We label the largest such subgraph G_χ. We use $\chi=|G_\chi|/|G_N|$ as a measure of congestion spread. We also compute connected subgraphs of nodes that are uncongested. We label the largest such subgraph G_α. We use $\alpha=|G_\alpha|/|G_N|$ as a measure of network connectivity.

	Enabled	Disabled
PD	buffers = 250×router speed	buffers = ∞
NC	3-tier 218-node topology as in Fig. 2 with routers labeled as core, PoP, D-class, F-class or N-class	flat 218-node topology as in Fig. 2 but with routers 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 & 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 *sst* = $2^{30}/2$) and congestion avoidance	packet transmissions not regulated by congestion-control

Table 3. Parameter values for each FxNS realism element

Packets injected into the network can be queued, dropped or delivered. We define effectiveness of packet delivery (π) as the ratio of delivered packets to injected packets. For each delivered packet we record the latency between injection and delivery times. We average these latencies over all delivered packets, and then normalize the average to fall between 0 (minimum delay) and 1 (maximum delay), yielding efficiency (δ) of packet delivery.

5. RESULTS AND DISCUSSION

For each combination simulated, we plotted each response (*y*-axis) vs. packet-injection rate (*x*-axis). Here we give plots for only the most abstract (c0) and realistic (c127) combinations, as discussed in Sec. 5.1. For each response, we also treat each of the 34 plots, one for each combination, as a 250-element vector and then cluster vectors to assess influence of each realism element on each response. We discuss the clusters in Secs. 5.2 to 5.5, drawing on insights

from the related *x*-*y* plots and multidimensional interactive visualization of FxNS simulation data [18]. All *x*-*y* and cluster plots are also available in an enlarged format [19].

5.1. Most Abstract vs. Most Realistic

Figure 4 contains four subplots comparing congestion behavior between the most abstract (c0) and realistic (c127) combinations. For combination c0, congestion spreads quickly with increasing packet-injection rate, encompassing all nodes by the time p reaches 500. For 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 c0 operate at the same speed. Core nodes become overwhelmed with congestion, which then spreads to the network edge. In c127, routers are engineered with varying, hierarchical speeds, so higher tiers can handle packet inflow rate from lower tiers. Second, c0 does not monitor and adapt to congestion, while c127 implements TCP, which measures congestion and adapts packet inflow-rate accordingly.

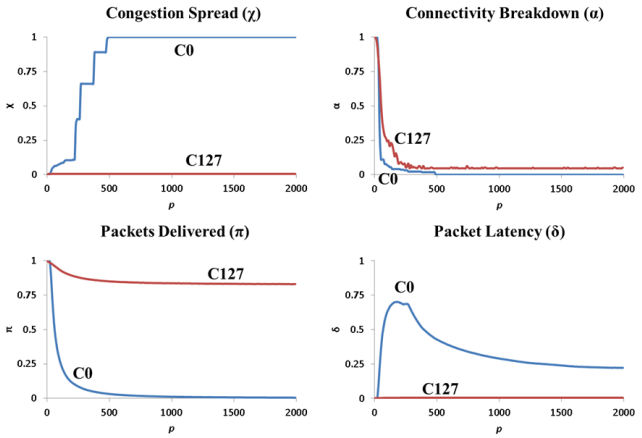

Figure 4. Comparing c0 vs. c127 for each response

Network connectivity breaks down quickly for both c0 and c127, reaching a low level as p passes 500. There are two main differences: c127 decays more slowly than c0 and c127 asymptotes with higher network connectivity. For c0 connectivity drops to zero after p passes 500. Combination c127 decays more slowly because TCP adapts packet injection based on measured congestion and c127 asymptotes with higher connectivity 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 core becomes congested and then congestion spreads to the edge, consuming all nodes.

For c0 proportion of packets delivered drops steeply, reaching nearly zero as p passes 1000. For c127 proportion of packets delivered drops modestly with increasing p, stabilizing near 80 %. This large difference arises from a combination of two factors: packet dropping and TCP. Combination c0 does not discard packets and does not adapt packet injection based on measured congestion. With increasing p, this causes a growing backlog of packets in all routers. Combination c127 discards packets when router buffers fill and adapts packet injection based on measured congestion. So undelivered packets for c127 encompass only discards, and rate adaptation limits their number.

For c127 latency of delivered packets remains low even as p increases beyond 2000. This occurs because packet dropping limits router queue sizes, so delivered packets are not long delayed. Without packet dropping, packet latency for c0 climbs steeply with increasing p, reaching an apex before decaying gradually. Delays climb because packet queues become jammed. Delays decay gradually because latencies are recorded only for delivered packets. At high p, c0 delivers relatively few packets, and those packets necessarily transit routes where queues are not jammed. Even with this decay, packet latency for c0 remains significantly above delay for c127.

5.2. Congestion Spread

Figure 5 shows hierarchical clustering for χ among all 34 combinations. Combination sequence numbers appear on the *x*-axis. The *y*-axis reports squared Euclidean distance. The plot indicates two main groups, separated by a large distance. The left-hand group contains combinations with VS or TCP or both enabled. These combinations show little congestion spread. Combinations in the right-hand group have VS and TCP disabled. These combinations show congestion spreading throughout the network.

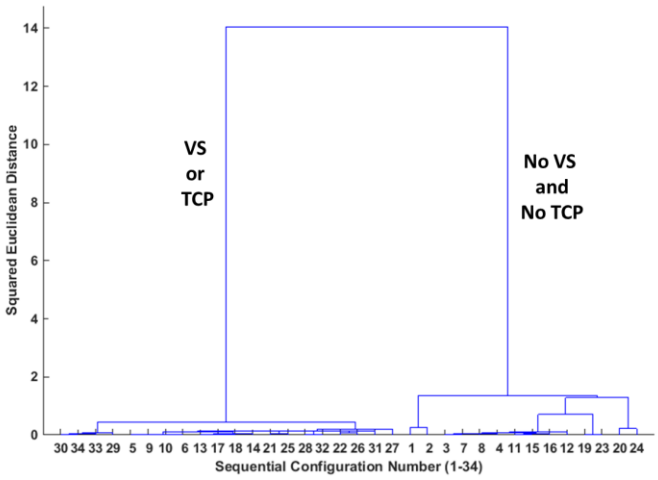

Figure 5. Clustering of congestion spread (χ)

5.3. Connectivity Breakdown

Figure 6 shows clustering for α. Note that distances among clusters in Fig. 6 are smaller than those in Fig. 5. This means connectivity breakdown is more similar among the combinations than is congestion spread. Breakdown in connectivity occurs when subgraphs of the topology are disconnected (due to congestion). As load increases connectivity breaks down even when congestion does not necessarily spread widely. Among combinations with VS disabled, the leftmost subgroup (sequence numbers 12, 15,

11, 3, 7, 8 and 16) in Fig. 6 has NC enabled. Our *x*-*y* plots show [19] these combinations reach complete breakdown sooner than others with VS disabled. With NC enabled, packet injection occurs at the network edge, thus packets flow in concentrated fashion to and through the network core. This differs from combinations c0 and c1 (sequence numbers 1 and 2), where packet injection can occur at any node, thus packet flow is more diffuse. Most configurations with VS disabled lost connectivity quickly and completely. Combinations with VS enabled and TCP disabled may experience complete connectivity breakdown, but the process requires higher packet-injection rates because more pressure must be applied from the edge before the core can congest. With both TCP and VS enabled, congestion stays at the edge.

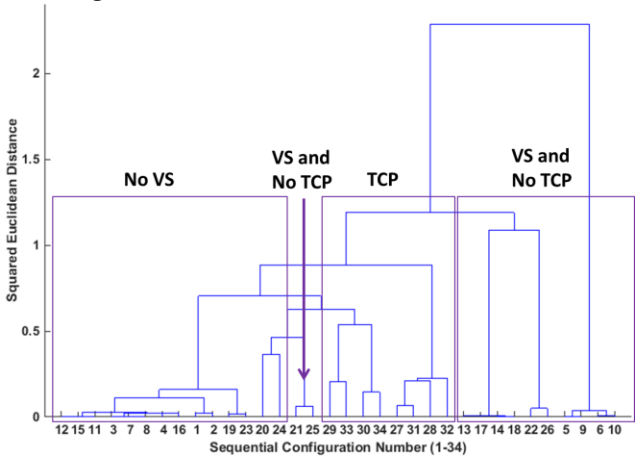

Figure 6. Clustering of breakdown in connectivity (**α**)

5.4. Packets Delivered

Figure 7 shows clustering for π. The plot indicates two main groups, separated by a large distance. The leftmost group contains combinations with TCP disabled, while the rightmost contains combinations with TCP enabled. The rate adaptation of TCP improves significantly the likelihood that an injected packet will reach the intended destination. Disabling TCP increases likelihood that an injected packet will be queued or discarded.

With TCP enabled, PD has a secondary influence on packet delivery. Disabling PD ensures that injected packets will be delivered eventually. But buildup of queues delays delivery, leading to timeouts and lower throughputs, as TCP reduces packet-injection rate. Enabling PD means some packets will be discarded, but TCP does not need to reduce injection rate as much. So throughputs remain higher, but likelihood of packet delivery decreases.

With TCP disabled, VS has secondary influence on packet delivery. Absence of VS allows queues to build widely among routers throughout a network. So, packets are more likely to be queued or discarded (depending on PD), and packet delivery approaches zero. With VS enabled packet queues build at the network edge. This reduces the number of routers where packets will be dropped or queued. In such cases, packet delivery approaches zero at a slower rate.

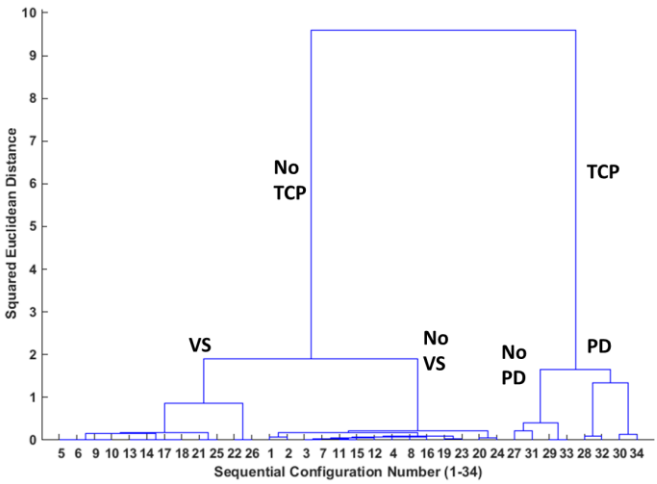

Figure 7. Clustering of packet delivery effectiveness (**π**)

5.5. Packet Latency

Figure 8 shows clustering for δ. We label the plot to show common factors in various groups and subgroups. With PD enabled, delivered packets experience little queuing delay, thus one-way latency is low. With PD disabled, packet queues become large with load, thus average one-way latency increases. With PD disabled, enabling TCP allows rate adaptation, thus buildup of large queues is less likely. This reduces delays for delivered packets. Enabling VS restricts large queues to routers at the network edge, which means that delivered packets have fewer large queues to transit. Disabling VS allows packet queues to form at any network router, which means delivered packets will have to transit through more large queues.

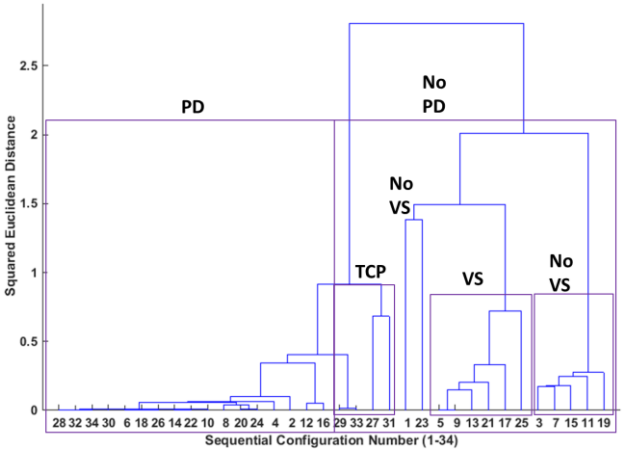

Figure 8. Clustering of packet delivery efficiency (δ)

5.6. Overall Findings

Realistic and abstract network models exhibit very different congestion behaviors. VS among router tiers, engineered to ensure adequate throughput, are very important to model. TCP, which detects congestion and adapts packet-injection

rate, is very important to model. PD from finite FIFO buffers is important to model for accurate measures of packet latency. Propagation delay (DE) is not important to model in networks spanning the continental US, but would be important in networks (e.g., interplanetary) where propagation delays may exceed queuing delays. A decade of studies [e.g., 3-12] used models too abstract to simulate realistic congestion in networks based on Internet technology. The validity of findings from such studies appears suspect.

6. CONCLUSION

We began with an abstract network simulation from the literature. We added realism elements in combinations, culminating with a high-fidelity simulation, also from the literature. By comparing patterns of congestion among the combinations, we showed that congestion spread in abstract models differs from congestion spread in realistic models. We described the influence of specific realism elements on congestion spread. We found that variable router speeds, the transmission-control protocol, and finite first-in, first-out buffers are important to model. We also found that propagation delay appears unimportant to model, when a simulated topology spans only the US. Finally, we demonstrated use of cluster analyses among response vectors to compare congestion spread, breakdown in connectivity and effectiveness and efficiency of packet delivery among a diverse set of network models.

We envision two directions for future work. First, we need to verify our findings for a variety of topologies, including interconnected networks. Second, we should explore whether random failures in the core, coupled with alternate routing, could cause cascading congestion. If so, we can seek precursor signals arising around the critical point. Such signals, if found, might provide warning of failure-induced congestion collapse.

ACKNOWLEDGMENTS

We appreciate financial support and encouragement by our laboratory management. We benefited from the review of our colleagues Guo Yang, Phil Gough, and Sandy Ressler, and also from anonymous, external reviewers.

REFERENCES

1. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M. and Hwang, D.-U. Complex networks: structure and dynamics, *Physics Reports*, 424 (2006), 175-308.
2. Stauffer, D. and Aharony, A. *Introduction to percolation theory: revised second edition*. Taylor & Francis, 1994.
3. Solé, R. and Valverde, S. Information transfer and phase transitions in a model of internet traffic, *Physica A*, 289 (2001), 595-605.
4. Woolf, M., Arrowsmith, D., Mondragon, R. and Pitts, J. Optimization and phase transitions in a chaotic model of data traffic, *Phys Rev E*, 66 (2002), 046106.
5. Arrowsmith, D., Mondragon, R., Pitts, J. and Woolf, M. Phase transitions in packet traffic on regular networks, ISSN 1103-467X, Institut Mittag-Leffler, 2004.
6. Mukherjee, G. and Manna, S. Phase transition in a directed traffic flow network, *Phys Rev E*, 71, 6 (2005), 066108.
7. Lawniczak, A., Lio, P., Xie, S. and Xu, J. Study of packet traffic fluctuations near phase transition point from free flow to congestion in data network model, in *Canadian Conference on Electrical and Computer Engineering*, (2007), 360-363.
8. Tadic, B., Rodgers, G. and Thurner, S. Transport on complex networks: flow, jamming and optimization, *International Journal of Bifurcation and Chaos*, 17, 7, (2007), 2363-2385.
9. Wang, D., Cai, N., Jing, Y. and Zhang, S. Phase transition in complex networks, *American Control Conference*, (2009), 3310-3313.
10. Rykalova, Y., Levitin, L. and Brower, R. Critical phenomena in discrete-time interconnection networks, *Physica A*, 389 (2010), 5259-5278.
11. Sarkar, S., Mukherjee, K., Ray, A., Srivastav, A. and Wettergren, T. Statistical mechanics-inspired modeling of heterogeneous packet transmission in communication networks, *IEEE Trans on Syst, Man, and Cybernetics—Part B: Cybernetics*, 42, 4 (2012), 1083-1094.
12. Echenique, P., Gomez-Gardenes, J. and Moreno, Y. Dynamics of jamming transitions in complex networks, *Europhys Lett*, 71, 2 (2005), 325.
13. Mills, K., Schwartz, E. and Yuan, J. How to model a TCP/IP network using only 20 parameters, *Winter Simulation Conference*, (2010), 849-860.
14. Dabrowski, C. and Mills, K. *The Influence of Realism on Congestion in Network Simulations*, NIST Technical Note 1905, January 2016, 62 pages. doi:10.6028/NIST.TN.1905. As of 5 Feb 2016.
15. Doyle, J., Alderson, D., Li, L., Low, S., Rougan, M., Shalunov, S., Tanaka, R. and Willinger, W. The "robust yet fragile" nature of the internet, *National Academy of Sciences*, 102, 41 (2005), 14497-14502.
16. Mills, K., Filliben, J., Cho, D., Schwartz, E. and Genin, D. *Study of proposed internet congestion control algorithms*, NIST SP 500-282, 2010.
17. Mills, K. and Filliben, J. Comparison of two dimension-reduction methods for network simulation models, *Journal of Research of the National Institute of Standards and Technology*, 116**,** 5 (2011), 771-783.
18. Gough, P., Multidimensional Interactive Visualization of FxNS Simulation Data. http://tinyurl.com/payglq6. As of 22 Oct 2015.
19. Dabrowski, C. and Mills, K. FxNS graphs enlarged. http://tinyurl.com/poylful. As of 15 Oct 2015.