# Evaluating Predictors of Congestion Collapse in Communication Networks

Christopher Dabrowski
*NIST*
Gaithersburg, MD, USA
cdabrowski@nist.gov

Kevin Mills
*NIST*
Gaithersburg, MD, USA
kmills@nist.gov

***Abstract*—Congestion in communication networks can be modeled as a percolation process, where congestion spreads minimally before a critical load and expands rapidly afterwards. Some studies identify precursor signals arising near critical load, but none attempt to predict congestion collapse. This paper investigates whether precursor signals, arising from changes in time series of router queue lengths, can predict onset of rapidly expanding congestion in time to alert network managers to take mitigating actions to avoid congestion collapse. The paper specifies five predictors: autocorrelation, variance, threshold, growth persistence, and growth rate. Predictor performance is measured for three simulated network models, under two traffic scenarios: increasing and steady load. Predictors are compared on implementation cost, accuracy, warning time, and persistence. The rates and types of prediction errors are also characterized. Results showed that: (1) predictor performance is influenced by network-model realism; (2) the autocorrelation and variance predictors performed poorly in some situations; (3) the threshold predictor yielded best overall accuracy, with mean warning time exceeding seven minutes for the most realistic network model. The paper also suggests a necessary condition to control false positives.**

***Keywords—network congestion, monitoring, and prediction***

## I. INTRODUCTION

Work to characterize and evaluate run-time predictors of congestion collapse in communication networks is timely and valuable, because simulation studies in several domains have found precursor signals arise around a critical point, marking a phase-transition boundary, e.g., between free-flowing traffic and widespread congestion. These developments appear promising as a theoretical basis for network monitoring methods that could be deployed to warn of impending congestion collapse.

Despite these foundational studies, no prior published work has evaluated the use of precursor signals to inform run-time predictors of incipient disasters in communication networks; such disasters include congestion collapse [1], considered in this paper, and cascading failures [15], not considered here. Automated, run-time methods to predict network disasters would enable alerting network managers to take remedial actions, forestalling total system failure. Effective run-time methods must be: practical to implement, sufficiently accurate, reliable, and timely. This paper evaluates those traits for five predictors, conceived to exploit measurable signals that could foreshadow congestion collapse.

Whereas previous studies of precursor signals use abstract network models, the current study adds two variants of a realistic network model, motivated by prior research [16], which found congestion spread in realistic models (e.g., bounded packet queues, tiered routers with properly engineered packet-forwarding rates, and congestion-control mechanisms) differs significantly from congestion spread in abstract models (e.g., unbounded packet queues, flat router topologies with uniform forwarding rates, and no congestion-control). These previous findings suggest signals near a critical point in an abstract network model might differ from signals in realistic models.

In the current study, an instance of each evaluated predictor runs independently in each of 218 routers. Each predictor instance analyzes time series of queue lengths in its own router and raises an alert whenever router overload is predicted. Predictors are evaluated in three network models under two traffic scenarios: increasing load and steady load. For each scenario, performance measures include proportion of routers where each predictor is consistent, i.e., signals onset of congestion before overload or raises no signal absent overload. Performance measures also include proportion of routers where each predictor is erroneous. Errors include: signaling incipient congestion absent overload, failing to signal when overload occurs, or signaling too late. Measures of consistent and erroneous predictions are combined to compute overall accuracy for each predictor. When predictors successfully forecast overload, additional performance measures are computed, including average warning time between first alert and overload, and average extent of signal persistence.

The paper makes four main contributions. First, the paper proposes five specific run-time predictors, and shows how to implement them in routers. Second, the paper defines an evaluation method to characterize and compare predictor performance. Third, the paper applies the evaluation method to characterize and assess the predictors. Finally, the paper shows that network-model realism influences predictor performance.

The remainder of the paper has five sections. Sec. II discusses related work. Sec. III outlines procedures needed to implement any predictor in a router; details each predictor; quantifies predictor implementation costs; and defines evaluation measures used to compare predictors in an experiment. As explained in Sec. IV, the experiment uses two realistic network model variants, described in Sec. IV.A, and one abstract network model, described in Sec. IV.B. Sec. IV.C details experiment traffic scenarios. Results are shown and discussed in Sec. V. Conclusions and future work appear in Sec. VI.

## II. Related Work

In some engineering and scientific domains, researchers attempt to predict onset of catastrophic events by detecting precursor signals arising around critical points preceding phase transitions. For example, Carreras et al. [4] defined thresholds based on relationships between power output and powerline capacity in an electrical grid. The risk of large-scale, cascading blackouts increased as the grid neared the thresholds. Similar work by Hines et al. [10] also found such thresholds, along with increased autocorrelation, but concluded that realistic grid models exhibited different threshold values than more abstract models. Climate scientists [9, 21] identified precursor signals when studying how autocorrelation and variance evolve in time series of climate-related measures. Such climate studies found that increases in autocorrelation and variance could be used as predictors of significant climate shifts.

Many researchers have explored evolution of congestion in simulations of communication networks. Some researchers [2, 8, 16] viewed evolving congestion to resemble a percolation process [24], where congestion spreads minimally prior to a critical load, and afterward spreads rapidly. Other researchers [13, 18-20, 22, 24-27] envisioned spreading congestion to exhibit a phase transition at critical load. Both types of studies identified precursor signals that arise around critical load. On the other hand, the studies used an array of abstract network topologies, generated by various random processes. The studies also considered many different packet-queue disciplines and routing strategies. The studies employed diverse measures of network congestion: one-way packet latency [18, 22, 25, 27]; packets delivered (i.e., aggregate throughput) [2, 18, 22]; queue lengths [2, 22, 25]; packets in transit [8, 13, 19, 26]; and packet drop rate [19]. Various studies analyzed the measures as time series, proportions, or variances. All these studies focused on characterizing how congestion measures evolve as a network model moves from an uncongested (free-flowing) state to a congested (overload) state.

No studies investigated explicitly the use of congestion measures as signals to predict network collapse. No studies assessed whether some signals were more reliable than others. No studies reported latency between first appearance of signals and transition to overload. No studies characterized how well congestion signals persist. Most studies used abstract network models, bearing little similarity to today's internet [5, 16].

This paper expands on previous studies by investigating whether congestion measures exhibit precursor signals that can be used to predict onset of congestion collapse. The paper considers the influence of realistic and abstract network models. The paper characterizes predictor implementation cost, accuracy, warning time, and persistence.

## III. Methods

This section describes procedures that can be implemented in routers to: generate, condition, and analyze time series; apply predictors; and raise alerts. The section then details five predictors, and their implementation costs. Finally, the section explains performance metrics used in subsequent sections to evaluate the predictors.

### A. Router Procedures

Each router generates a time series of queue lengths within a sliding window, uses some predictor to analyze the time series, and raises an alert whenever the predictor exceeds a designated limit. Routers could timestamp and forward such alerts to a network control center, which can integrate and analyze alerts from all routers, giving a network-wide picture of spreading congestion. The goal of any predictor is to issue alerts that precede router overload, thus providing warning time for network managers to take remedial actions.

To generate a time series of queue lengths, each router periodically samples the count of queued packets and computes some statistic within a fixed time slot. The sampling period should be chosen to yield reasonable resolution without unduly burdening routers. In this paper, samples are taken every 200 ms in a 10-s time slot, so each data point in a time series is the average (or variance in) queue length over 50 samples (i.e., 5 samples/s). Fig. 1 gives an example raw time series (jagged curve) of averaged queue lengths over 3300 s, sampled from a router in a simulated network under increasing load.

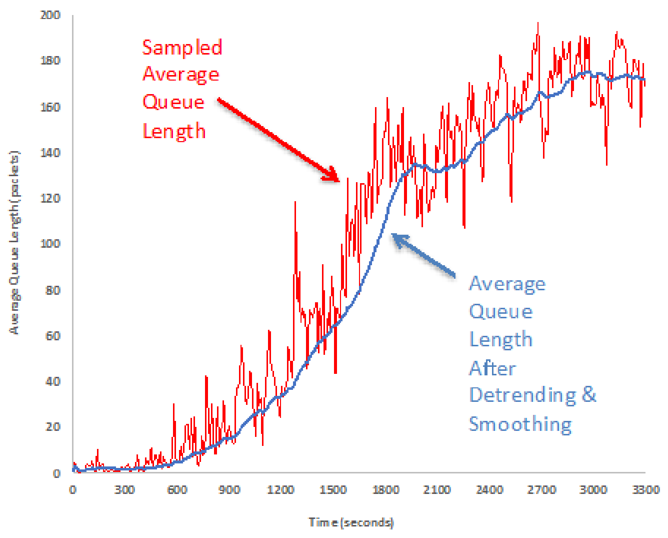

Fig. 1. Example raw (jagged curve) and smoothed time series

Raw samples are smoothed to reduce time-series noise, and create a more slowly changing curve. Among many possible methods, this paper uses Nadaraya-Watson smoothing [9], also used by other researchers [7] to remove noise from historical measurements of Earth's climate history. The resulting curve is detrended to remove the effects of normal traffic, allowing predictors to focus on any changes in queue lengths due to atypical traffic. This paper implements detrending by subtracting an appropriate queue-length statistic (i.e., mean or variance), as measured for each router under normal traffic. Actual networks may include more complex trends than were present in these simulations, thus additional detrending may be required in such situations. Fig. 1 shows results (smooth curve) from smoothing and detrending a time series (jagged curve) for one router in a simulation experiment from this paper.

Predictors analyze a sliding window over a detrended and smoothed time series of sampled queue-length statistics. This paper uses a 300-s window, i.e., each window contains 30 10-s time slots looking back in time and slides forward every 10 s. The 300-s window provides a reasonable span of history to consider, and a 10-s advance provides for timely generation of alerts. Of course, other choices for window size and advance increment are possible, and could yield results that differ from those reported here. (Ongoing work is exploring the influence of varying values for all possible system parameters.)

While each predictor operates differently in several details (explained below), all predictors use an identical procedure, shown in eq. 1, which is implemented in each router. Given a window of $W$ time slots, a router counts the number of slots where a predictor metric is ≥ a threshold, and then raises an alert if that count is ≥ a limit within the window. Specifically,

$$\sum_{i=1}^{W}[\mathrm{P}(Q_i) \geq P_{Threshold}] \geq P_{AlertLimit}\,, \tag{1}$$

where $W$ is sliding window size in slots, P denotes a predictor function, $Q_i$ is the detrended and smoothed queue-length statistic for the $i$th slot in a window, $P_{Threshold}$ is a predictor's threshold, and $P_{AlertLimit}$ is a predictor's alert limit within a sliding window. Note the use of Iverson's brackets [11] within the summation, which evaluates to 1 if the proposition is true, but 0 otherwise. As explained next, each predictor may differ in implementation of P, $P_{Threshold}$, and $P_{AlertLimit}$. While only five predictors were implemented and evaluated for this paper, the general framework permits predictors to be added easily.

## B. Predictor Specifications

This paper investigates five predictors: autocorrelation, variance, threshold, growth persistence, and growth rate. As explained below, for each predictor, specific values were selected for $P_{Threshold}$ and $P_{AlertLimit}$. The values chosen appeared pragmatic for the network models simulated here. Different values might be needed to yield effective results for other network models. Each predictor is denoted as a triple, (P, $P_{Threshold}$, $P_{AlertLimit}$), and described in turn.

***Autocorrelation*** (α(1), 0.7, 21). This predictor is motivated by research [5, 10, 20] in several domains that suggests autocorrelation increases near a critical point around a phase transition. Here, α(1) is a predictor function that computes lag-1 autocorrelation for average (detrended and smoothed) queue lengths, $Q$, in each sliding window. Signal strength diminished for greater lags.

Positive autocorrelation might be judged significant within the range of 0.5-1.0. Here, $P_{Threshold}$ is assigned as 0.7 because that value is moderately significant, and should provide for earlier signals than larger values. Regarding $P_{AlertLimit}$, several values were considered, including 1, 10, and 21 slots in the sliding window. The lower values, while signaling earlier, caused excessive false alerts, thus $P_{AlertLimit}$ was set to 21. The autocorrelation predictor alerts whenever α(1) ≥ 0.7 for 21 (or more) of the 30 slots in a sliding window.

The α(1) function used the MATLAB xcorr() algorithm [28] to compute cross-correlation between the $Q$ and lag-1 $Q$ time series. This approach is consistent with computing statistical lag-1 autocorrelation using

$$\alpha(1) = \frac{\mathrm{E}[(Q_i-\mu(Q))(Q_{i-1}-\mu(Q))]}{\sigma^2(Q)}, \tag{2}$$

where E[] is expected value, μ($Q$) is queue length averaged across the sliding window, $\sigma^2(Q)$ is the variance in queue length across the sliding window, $Q_i$ is average queue length in the $i$th slot, and $Q_{i-1}$ is average queue length for the prior slot.

While the results in Sec. V were obtained using xcorr(), which converts time-series data using Fast-Fourier transforms, results were also computed for two other formulations of α(1): MATLAB xcov() and statistical autocorrelation, as derived by a simulation-package function. These alternate formulations produced results like those obtained with xcorr().

***Variance*** (ν, μ(ν)+|3σ(ν)| : *steady load*, 21). This predictor is motivated by research [9, 21] that suggests increases in variance may foreshadow incipient phase transition. For this predictor, the sliding window advances over variances (not averages) in queue length. The variance (ν) in queue length for each 10-s slot in a sliding window is computed using eq. 3,

$$\nu = \frac{\sum_{i=1}^{N}|Q_i - \mu(Q)|^2}{(N-1)}, \tag{3}$$

where $N$ is the number (i.e., 50) of 200-ms samples in a slot, $Q_i$ is queue length for the $i$th sample, and μ($Q$) is the mean queue length across all samples in the slot. $P_{Threshold}$ is assigned as μ(ν) + |3σ(ν)|, where μ(ν) is the mean variance and σ(ν) is the standard deviation in variance under normal traffic (i.e., steady load). Variance beyond $P_{Threshold}$ under normal traffic should occur by chance less than 1% of the time. For reasons given earlier, $P_{AlertLimit}$ is set to 21. The variance predictor alerts whenever ν ≥ $P_{Threshold}$ for at least 21/30 slots in the sliding window.

Some prior research [18] in communications networks suggests that variance in queue lengths decreases as a system becomes congested. For that reason, an alternate version of the variance predictor was implemented by reversing the first inequality in eq. 1 to < and changing $P_{Threshold}$ to μ(ν)+|σ(ν)|. Those results, not reported here, showed that decreases in variance did fall below $P_{Threshold}$ (and approached zero) as router buffers filled, but the decreases occurred close to (or after) router queues reached overload, and so provided little (or no) warning. In other words, variance did decrease as routers congested, but the decrease occurred too late for prediction.

***Threshold*** ($Q_i$, 0.25($C$ - μ($Q$) : *steady load*), 1). This predictor, motivated by simplicity, generates an alert whenever the average queue length, $Q_i$, ≥ $P_{Threshold}$. $P_{Threshold}$ is set to a fraction (0.25) of the number of normally empty packet buffers, as computed by subtracting the mean queue length, μ($Q$), under normal traffic (i.e., steady load) from buffer capacity, $C$. $P_{Threshold}$ was set to 0.25, which is about 2/3 below overload (set to 0.7, as explained in Sec. III.D), and represents a significant increase (about 1/3) in queue length towards overload, yet allows for ample warning times. The threshold predictor alerts whenever queued packets consume 25% or more of normally empty buffers.

***Growth Persistence*** ($Q_i \mid Q_i > Q_{i\text{-}1}$, 0.25($C$ - μ($Q$) : *steady load*), 21). This predictor is motivated by the possibility that the threshold predictor might be too crisp, leading to false alerts. The growth-persistence predictor introduces two refinements. First, the function returns average queue length, $Q_i$, to compare with $P_{Threshold}$ only when $Q_i > Q_{i\text{-}1}$; otherwise, 0 is returned. This ensures that average queue length surpasses $P_{Threshold}$, chosen as before, and grows over time. Second, $P_{AlertLimit}$ = 21, chosen as explained previously, ensures that queue-length increases persist in time. These refinements mean that the growth-persistence predictor alerts only when queue length ≥ $P_{Threshold}$, while also increasing over time, i.e., for at least 21/30 slots in a sliding window.

***Growth Rate*** ((estimated $Q_{j\text{-}1}$ + $s_{j\text{-}1}$) + $s_j$, 0.25($C$ - μ($Q$) : *steady load*), 21). This predictor stems from an idea that estimating queue length based on the growth rate of the real queue might lead to quicker and more persistent alerts. The predictor computes an estimated queue length for each window $j$ by adding a slope, $s_j$, to the sum of the estimated queue length and slope from the previous window $j$-1. Slope, $s$, is computed using a least-squares method, as given in eq. 4,

$$s = \frac{\sum_{i=1}^{W}(i-(W+1)/2)\big(Q_i-\mu(Q)\big)}{\sum_{i=1}^{W}(i-(W+1)/2)^2}, \qquad (4)$$

where $W$ is window size, $Q_i$ is queue length of the $i$th slot in the current window and μ($Q$) is the mean queue length across the current window. The values for $P_{Threshold}$ and $P_{AlertLimit}$ were chosen as explained previously. The growth-rate predictor alerts only when estimated queue length ≥ $P_{Threshold}$ for at least 21/30 slots in the sliding window.

## C. Predictor Implementation Costs

Table I gives total cost to implement each predictor in units of thousand processor cycles (kilocycles) per sliding window. Costs are also decomposed by algorithm element: (1) sample and detrend, (2) smooth, (3) compute predictor, and (4) decide alert. Executing the predictors on a 3.4 GHz processor takes from 11.9 us (TH) to 23.3 us (AC) for each sliding window.

TABLE I. PROCESSOR KILOCYCLES/WINDOW (30 10-S SLOTS) FOR EACH PREDICTOR: AC=AUTOCORRELATION; VR=VARIANCE; TH=THRESHOLD; GP=GROWTH PERSISTENCE; GR=GROWTH RATE; ↓=LOWER IS BETTER; BEST COSTS IN **BOLD**

| Predictor | Sample/ Detrend | Smooth | Compute Predictor | Decide Alert | Total↓ |
|---|---|---|---|---|---|
| AC | **0.98** | **25.86** | 51.90 | 0.41 | 79.14 |
| VR | 36.10 | 26.58 | 3.21 | **0.36** | 66.26 |
| TH | **0.98** | **25.86** | **0** | 13.61 | **40.45** |
| GP | **0.98** | **25.86** | **0** | 13.95 | 40.78 |
| GR | **0.98** | **25.86** | 1.19 | 12.93 | 40.95 |

## D. Performance Metrics

For experiment purposes, each simulated router raises an overload exception (O) when a packet queue ≥ 70% of buffer capacity. This value corresponds roughly to the minimum utilization in a queuing system beyond which delays could begin to grow without bound [23]. Any given predictor may issue an alert (A) of oncoming router overload. When overload and alert occur, given that the alert appears before overload (i.e., $A_t < O_t$, where $t$ is time), then a predictor is successful. Consistent predictor outcomes include: true positive (*tp*: $A_t < O_t$) and true negative (*tn*: ¬A ∧ ¬O). A predictor is erroneous under any of three conditions: false positive (*fp*: A ∧ ¬O), false negative (*fn*: ¬A ∧ O), or late positive (*lp*: $A_t > O_t$).

Given these definitions, predictor accuracy is defined as: (*tp* + *tn*) / (*tp* + *tn* + *fp* + *fn* + *lp*). For true positives, one can measure warning time as latency ($O_t$ - $A_t$) between first alert and overload. One can also measure persistence as proportion of the latency interval over which an alert remains asserted. Computing persistence is possible because predictors evaluate conditions for an alert with every advance in a sliding window.

## IV. EXPERIMENT

Each of the five predictors described in Sec. III.B was compared under six situations, paired combinations of three, simulated network models and two traffic scenarios. This led to (5 x 3 x 2 =) 30 configurations, where each configuration draws one element from each column (predictor, network model, traffic scenario) in Table II.

TABLE II. PREDICTOR TEST CONFIGURATIONS

| Predictor | | Network Model | | Traffic Scenario |
|---|---|---|---|---|
| Autocorrelation | | Realistic TCP | | Increasing Load |
| Variance | | | | |
| Threshold | X | Realistic UDP | X | |
| Growth Persistence | | | | Steady Load |
| Growth Rate | | Abstract | | |

For each configuration, consistent (i.e., *tp* and *tn*) and erroneous (i.e., *fp*, *fn* and *lp*) outcomes were tabulated for each of 218 routers, at the end of 3000 simulated seconds (covering 300 sliding windows). Subsequently, the measures described in Sec. III.D were computed across all routers, yielding the results reported in Sec. V. The remainder of Sec. IV describes the two realistic (A) and one abstract (B) network models simulated, as well as the two traffic scenarios (C).

## A. Realistic TCP and UDP Network Models

Many studies of network congestion adopt abstract models, which include unrealistic assumptions. While using such an abstract model, this study also uses two realistic network model variants: transmission-control protocol (TCP) and user-datagram protocol (UDP). Both variants share common features: topology, node-speed assignments, buffer-sizing algorithm, and flow-injection and management procedures. The two variants differ only in packet-injection procedures. First, the common features are explained, followed by the distinct packet-injection procedures.

***Topology***. The realistic network models, drawn from previous work (see [16-17] for further details), use a four-tier topology adapted from an internet-service provider network. The topology, illustrated in Fig. 2, contains 218 routers organized as three tiers: (1) 16 core (A-P), (2) 32 point-of-presence, or PoP (A1-P2), and (3) 170 edge (A1a-P2g). The topology also includes a fourth tier (not shown) that contains 51 588 source nodes and 206 352 receiver nodes, spread evenly below the edge routers, which brings the number of network nodes to just over a quarter million (258 158).

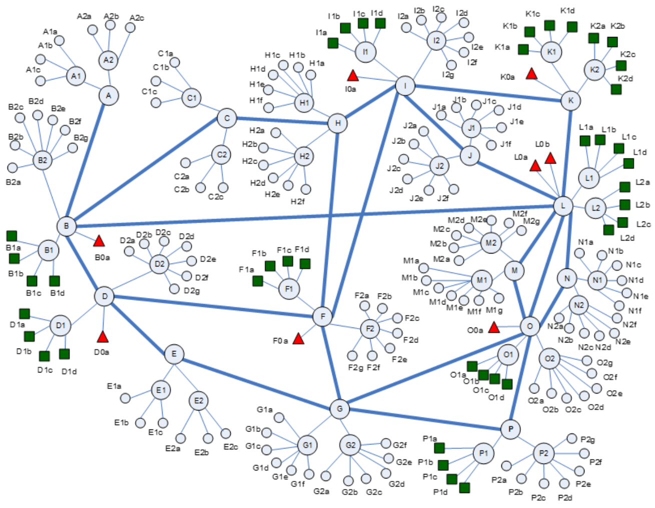

Fig. 2. Three-tier 218-router topology – 16 core (A-P), 32 PoP (A1-P2) and 170 edge (A1a-P2g)

In the model, routers forward each outgoing packet to a next router chosen based on a fixed, shortest path toward the destination. Shortest path is determined by the minimum count of links to a given destination.

***Node-Speed Assignments***. Router tiers in the topology were assigned fixed relationships among forwarding speeds, as shown in Table III, to reflect sound network-engineering principles. Forwarding speeds of all routers can be adjusted with a single parameter, *S*, while maintaining fixed speed relationships. For this study, *S* was set to 40 packets/ms, which determined all router forwarding speeds.

TABLE III. DEFINED RELATIONSHIPS AMONG TIERED ROUTER FORWARDING SPEEDS

| Tier | Speed |
|---|---|
| Core | 2*S* |
| PoP | *S*/4 |
| Edge: Normal | *S*/4/10 |
| Edge: Fast | 2*S*/4/10 |
| Edge: Very Fast | *S*/4 |

Fig. 2 denotes forwarding speeds for the 170 edge routers with three colors/shapes: gray/circle (122 normal), green/square (40 fast), and red/triangle (eight very fast). Setting *S* to 40 assigned forwarding speeds as: 80 packets/ms for core routers; 10 packets/ms for PoP and very fast edge routers; 2 packets/ms for fast edge routers; and 1 packet/ms for normal edge routers. Speeds were also assigned to source and receiver nodes, where half (randomly selected, uniform) operate at 2 packets/ms and half at 0.2 packets/ms. While edge routers may become overloaded due to traffic injected by sources and receivers, the defined speed relationships among router tiers assure that forwarding speeds for core and PoP routers are congruent with the maximum possible incoming traffic from connected routers.

***Buffer-Sizing Algorithm***. Each router is assigned a finite buffer capacity, computed as **ceil** [250 ms × router speed] packets, following the Bush-Meyer guidelines [3], as suggested for the internet. Core routers forward packets from a single buffer, while PoP and edge routers have two (half-sized) buffers, one heading "in" toward the core and one heading "out" toward the edge. For PoP and edge routers, packet forwarding alternates between "in" and "out" buffers. Packets arriving at a buffer are queued first-in-first-out for forwarding. Packets arriving at a full buffer are discarded.

***Flow-Injection and Management Procedures***. At randomly selected times, varying with traffic load, sources inject correlated flows of packets at their parent edge router. To simulate heavy-tailed flows often seen in internet traffic, flow sizes were chosen from a Pareto distribution with a mean of 350 packets and shape of 1.5. Traffic load varies inversely with mean time between flow arrivals, which is controlled by packet-injection rate, *p*. At any given time, any idle source injects a flow with probability $p/s \times f$, where *s* is the number of sources and *f* is mean flow size.

To inject a flow, a source: (1) selects a random (uniformly distributed) receiver from under a core router different from the source's core, (2) sends (and resends if necessary) a connection request until receiving connection acceptance from the chosen receiver, (3) selects a flow size, and (4) follows one of the packet-injection procedures, explained next. After completing the packet-injection procedures, a source becomes idle until its next chosen flow-injection time.

***TCP Packet-Injection Procedures***. In the TCP variant of the realistic network model, sources inject the packets of a flow at a controlled rate determined by congestion-control and reliable-transmission procedures: (1) slow-start, (2) congestion-avoidance, (3) retransmission, and (4) timeouts. These TCP procedures determine when a source may send packets toward a receiver. At any given time, a source may send a prescribed number of packets (known as congestion window, or *cwnd*) prior to receiving feedback from the receiver. Thus, *cwnd* controls the rate of packet transmission on a flow.

Using TCP slow-start procedures, a source increases a flow's *cwnd* exponentially from a small initial value (two packets, here) until a lost packet is detected or *cwnd* reaches a threshold, known as logarithmic slow-start threshold, or *log-sst* (set to 100 packets, here). If *cwnd* passes *log-sst* without packet loss, then a source increases *cwnd* logarithmically until reaching another slow-start threshold, or *sst* (set to $2^{30}/2$ packets, here). If *cwnd* reaches *sst*, a source enters congestion-avoidance, subsequently increasing *cwnd* more slowly, at a linear rate. If a packet is lost, *cwnd* is reduced in half and then increased linearly until another packet is lost, after which *cwnd* is reduced in half again, and so on. This algorithm leads to a saw-tooth pattern in *cwnd*, inducing a corresponding variation in a flow's transmission rate.

For every data packet sent by a source, receiver feedback is expected: either an *ack* (acknowledgment) or *nak* (negative *ack*). Any packet, whether data, *ack*, or *nak*, may be discarded at routers with full buffers. A receiver sends a *nak* whenever a data packet fails to arrive in the expected sequence from the source. For every *nak* received by a source, a data packet must be retransmitted. When a source fails to receive any *ack* or *nak* for a prolonged period, i.e., a timeout, then *sst* is reset to the current *cwnd* and the source reenters slow-start. These procedures continue until every data packet (including any retransmissions) sent by a source is confirmed by the receiver.

***UDP Packet-Injection Procedures***. In the UDP variant of the realistic network model, sources inject the packets of a flow at an assigned rate (2 or 0.2 packets/ms). Once all packets are injected, the flow becomes idle. No congestion-control or reliable-transmission procedures are used.

## B. *Abstract Network Model*

The abstract network model is taken from previous work (see [8, 16] for details). The topology shown in Fig. 2 is flattened by removing router tiers so that all 218 routers are peers. The fourth tier of sources and receivers is removed. Each router is provisioned with a single, infinite buffer. Traffic is injected as individual packets at rate $p$, which is varied to modulate load. Each injected packet is added to the end of the forwarding queue of a randomly (uniform) chosen source router, and assigned a randomly (uniform) chosen destination router, other than the source. All routers are assigned the same forwarding speed (9 packets/ms), computed as the weighted average speed across all routers in the realistic models. Queued packets are forwarded to a next router based on the minimum count of links to the destination. The abstract model uses no congestion-control procedures.

## C. *Traffic Scenarios*

This study simulated two distinct traffic scenarios: steady load and increasing load. Both scenarios begin with a warmup period where $p$ = 10 packets/s for 300 s. Subsequently, a measurement period of 3000 s is simulated. For the steady-load scenario, $p$ = 10 for the entire period. For the increasing-load scenario, $p$ is increased by 10 packets/s every 10 s, reaching $p$ = 300 packets/s by the end of the 3000-s measurement period.

Experiment simulations encompassed six situations: each a paired combination of network model and traffic scenario. During the 3000-s measurement period, each simulation produced a queue-length time series (200-ms samples) for each router. Each simulation was repeated for 10 repetitions, with different random-number seeds, and the resulting queue-length time series were averaged over the repetitions to produce a time series per router for each of the six situations. This means that a set of six, queue-length time series, one per combination of network model and traffic scenario, were averaged over ten simulations to create the time series seen by each router.

For each time series, a predictor-appropriate statistic (mean or variance) was computed from 200-ms samples for each 10-s slot, and the time series was detrended and smoothed. Each router applied five predictors to the set of time series covering the six situations, each representing a paired combination of network model and traffic scenario. Taking these steps created a fair test, eliminating the possibility for random variation in time series to influence predictor performance.

## V. Results and Discussion

Table IV reports consistent (true positive and true negative) outcomes for the five predictors, under the increasing load scenario, in each of the three simulated network models. Before discussing the results, a measurement detail must be resolved, regarding alerts issued by PoP and core routers in the realistic network models. Recall that PoP and core routers are engineered to handle maximum expected load, and thus do not experience overload. Even so, predictors in PoP and core routers can issue alerts. How should such alerts be classified? One could declare those alerts erroneous, or could exclude PoP and core routers from the results. On the other hand, if subordinate routers alerted successfully, there would be justification for the parent (PoP or core) router also alerting. The results in Table IV include PoP-router alerts in true positives when all subordinate edge routers also issued successful alerts ($A_t < O_t$). Similarly, core-router alerts are included in true positives when both subordinate PoP routers also issued true-positive alerts.

TABLE IV. Successful Outcomes under Increasing Load: TPR=True-Positive Rate; TNR=True-Negative Rate; ↑=Higher is Better; Con=Consistency; TCP=Realistic TCP Model; UDP=Realistic UDP; ABS=Abstract Model

| | TPR ↑ | | | TNR ↑ | | | Con (TPR+TNR) | | |
|---|---|---|---|---|---|---|---|---|---|
| | TCP | UDP | ABS | TCP | UDP | ABS | TCP | UDP | ABS |
| AC | 0.913 | 0.963 | 0.131 | 0 | 0 | 0.805 | 0.913 | 0.963 | 0.936 |
| VR | 0.862 | 0.963 | 0.133 | 0 | 0 | 0 | 0.862 | 0.963 | 0.133 |
| TH | 0.723 | 0.75 | 0.128 | 0.254 | 0.25 | 0.859 | 0.977 | 1 | 0.986 |
| GP | 0.634 | 0.74 | 0 | 0.259 | 0.26 | 0.853 | 0.894 | 1 | 0.853 |
| GR | 0.686 | 0.75 | 0 | 0.254 | 0.25 | 0.853 | 0.94 | 1 | 0.853 |

The last column (Con) in Table IV reports the proportions of (218) routers for which predictors exhibited consistent alert behavior under increasing load for each network model. Consistent behavior sums true positives (TPR↑ column) and true negatives (TPN↑ column). Results for all predictors, except variance, are reasonably good. Causes for erroneous predictions are shown below in Table VI.

In Table IV true negatives ranged between 81-86% in the abstract network sub-column (ABS) for all but variance. (Note that the variance predictor does not report these true negatives for the abstract network model, but instead raises erroneous false positives, as discussed below when considering Table VI.) The high rate of true negatives in the abstract network model implies that about 85% of routers fail to reach overload. This occurs due to a combination of three main factors: load is diffused among all routers, router forwarding speeds are homogeneous within a flat topology, and some routers have high centrality.

Diffused load arises as individual packets are injected uniformly across all routers. But every injected packet must travel on some route from source to destination. Visualization [6] of spreading congestion in the abstract model showed the overloaded 15% of routers exhibit high centrality, i.e., many shortest-path routes transit them, and thus transit packets are forwarded through them. Given that topology is flat in the abstract network model and that all routers have identical forwarding speeds, high-centrality routers have insufficient forwarding capacity to cope with the increased transit traffic. The uncongested 85% of routers exhibit low centrality, receiving mainly injected packets. In the realistic network models with a tiered topology, core routers have high centrality but are assigned higher forwarding speeds to compensate for increased transit traffic, and thus those routers do not congest so easily.

For each predictor, Table V reports mean warning time (i.e., latency) for true-positive alerts under each network model. The distribution (not given here) of latencies for the TCP model showed warning times exceeded 15 min under each predictor for a significant proportion of true-positive routers: 22% (AC), 23% (VR), 16% (TH), 12% (GP), and 15% (GR). Those proportions were even higher for the UDP model. For the abstract network model, only autocorrelation (61%) and variance (65%) gave warning times exceeding 15 min. Alerts persisted well for all predictors.

TABLE V. MEAN WARNING TIME (LATENCY) AND ALERT PERSISTENCE FOR TRUE POSITIVES (TPR COLUMN) FROM TABLE IV

| | Mean Latency (min) ↑ | | | Mean Persistence↑ | | |
|---|---|---|---|---|---|---|
| | TCP | UDP | ABS | TCP | UDP | ABS |
| AC | 10.73 | 22.89 | 17.52 | 1 | 1 | 0.992 |
| VR | 10.48 | 21.72 | 24.33 | 0.994 | 1 | 0.991 |
| TH | 7.67 | 15.77 | 0.58 | 1 | 1 | 1 |
| GP | 4.61 | 11.86 | — | 0.972 | 0.99 | — |
| GR | 5.63 | 13.53 | — | 1 | 1 | — |

Autocorrelation and variance gave most warning time because the other predictors transitioned more slowly to alerts, especially under realistic UDP and abstract network models. Slower transition occurs because the other predictors must wait for packets to occupy at least 25% of normally empty packet buffers, while autocorrelation and variance do not include this requirement in $P_{Threshold}$. Among the other predictors, threshold raises an alert after a single crossing of $P_{Threshold}$, thus the predictor gives more warning time than growth persistence and growth rate, which require crossing $P_{Threshold}$ in 21/30 slots.

The realistic TCP network model yields lower warning times overall across the predictors because TCP congestion-control procedures slow the rate of congestion increase, which causes predictor measures to transition more slowly toward an alert state. In other words, gradual increase in congestion limits the warning time that can be achieved by any predictor.

For each predictor and network model under increasing load, Table VI gives the error rate for the residual proportions (i.e., 1 – Con) of routers not represented in Table IV. Across all network models, the threshold predictor had lowest mean error rate of 1.2%. Somewhat higher mean error rates occurred for autocorrelation (6%), growth persistence (8.4%), and growth rate (6.9%). The variance predictor had an unacceptably high mean error rate (34.7%), dominated by an 86.7% error rate under the abstract network model (mainly false positives). The variance predictor signaled false alerts in the abstract network model because variance in queue lengths under normal traffic was quite low, so with increasing load the variance predictor could exceed $P_{Threshold}$ even when the absolute number of queued packets was small. For this reason, the variance predictor is unreliable.

TABLE VI. ERROR RATE (1-CON) UNDER INCREASING LOAD –ERROR RATE IS DECOMPOSED BY ERROR TYPE IN TABLE VII

| | Error Rate (1 – Con) ↓ | | | Mean Error Rate |
|---|---|---|---|---|
| | TCP | UDP | ABS | |
| AC | 0.087 | 0.037 | 0.074 | 0.06 |
| VR | 0.138 | 0.037 | 0.867 | 0.347 |
| TH | 0.023 | 0 | 0.014 | 0.012 |
| GP | 0.106 | 0 | 0.147 | 0.084 |
| GR | 0.06 | 0 | 0.147 | 0.069 |

Table VII decomposes the error rates from Table VI by type: alerting absent overload (*fp*: A ∧ ¬O); failing to alert in presence of overload (*fn*: ¬A ∧ O); alerting too late (*lp*: $A_t > O_t$). No predictors made false-negative errors. For all models, the autocorrelation and variance predictors made mainly false-positive errors. The growth-persistence and growth-rate predictors erred mainly by alerting too late for the abstract and TCP network models. The threshold predictor, which made few errors, sometimes alerted too late in the abstract network model. For the realistic UDP model, the threshold, growth-persistence, and growth-rate predictors made no errors.

TABLE VII. ERROR RATE UNDER INCREASING LOAD FROM TABLE VI, DECOMPOSED BY ERROR TYPE: FPR=FALSE-POSITIVE RATE; FNR=FALSE-NEGATIVE RATE; LPR= LATE-POSITIVE RATE

| | FPR ↓ | | | FNR ↓ | | | LPR↓ | | |
|---|---|---|---|---|---|---|---|---|---|
| | TCP | UDP | ABS | TCP | UDP | ABS | TCP | UDP | ABS |
| AC | 0.087 | 0.037 | 0.058 | 0 | 0 | 0 | 0 | 0 | 0.016 |
| VR | 0.12 | 0.037 | 0.85 | 0 | 0 | 0 | 0.018 | 0 | 0.017 |
| TH | 0.023 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.014 |
| GP | 0.023 | 0 | 0 | 0 | 0 | 0 | 0.083 | 0 | 0.147 |
| GR | 0.023 | 0 | 0 | 0 | 0 | 0 | 0.037 | 0 | 0.147 |

Three predictors (threshold, growth persistence, and growth rate) had a 2.3% error rate for the TCP network model, i.e., issued false positives for five of 218 routers. These five routers, comprising the entire 2.3% error rate, were (fast) edge routers that alerted absent an overload (A ∧ ¬O). In those cases, router queue lengths had reached at least 64% of buffer capacity, and would have reached overload given more simulated time.

For the steady-load scenario, Table VIII reports false-positive (A ∧ ¬O) rates for the predictors by network model. The threshold, growth-persistence, and growth-rate predictors issued no false alerts under steady load. For realistic network models, the autocorrelation predictor issued predominantly false alerts. This occurs because autocorrelation measures self-similarity in queue lengths, which were highly self-similar (autocorrelation just below one) under the realistic models, where packet injection occurs in correlated streams at edge routers. The autocorrelation predictor had few false alerts for the abstract network model because diffuse packet injection led only to small, sporadic, queues in routers with low centrality, i.e., most routers. Autocorrelation was set to zero when a router queue is empty, leading to true negatives in those cases.

TABLE VIII. FALSE-POSITIVE RATES UNDER STEADY LOAD

| | FPR ↓ | | |
|---|---|---|---|
| | TCP | UDP | ABS |
| AC | 0.995 | 0.927 | 0.064 |
| VR | 0.362 | 0.45 | 0.949 |
| TH | 0 | 0 | 0 |
| GP | 0 | 0 | 0 |
| GR | 0 | 0 | 0 |

While the variance predictor issued fewer false alerts for the realistic network models, the false-positive rate was still quite high (36-45%). The variance predictor had fewer false alerts for the realistic network models because the standard deviation in variance was higher under normal traffic for those models, and thus $P_{Threshold}$ was higher, reducing the number of false alerts. High false-positive rates under steady load in the real network models disqualify both the autocorrelation and variance predictors from further consideration for practical use.

Additional experimentation discovered that false alerts under steady load could be eliminated for the autocorrelation and variance predictors by adding a buffer threshold condition, $0.25(C - \mu(Q))$ : *steady load*), to the $P_{Threshold}$ values. Under increasing load for realistic networks, the cost of adding the condition was diminished alerting accuracy and decreased warning time. These alternate results (not shown here) suggest that including a buffer-capacity to queue-length relationship in $P_{Threshold}$ values is necessary to control false alerts. That is, any predictor signal should be supported by evidence that router queue lengths are elevated significantly above levels measured under normal traffic. The tradeoff, though, is to reduce warning time, and to increase (slightly) false-negative rate.

TABLE IX. OVERALL PREDICTOR ACCURACY (BEST FOR EACH NETWORK MODEL IN **BOLD**)

| | Accuracy ↑ | | |
|---|---|---|---|
| | TCP | UDP | ABS |
| AC | 0.459 | 0.518 | 0.936 |
| VR | 0.75 | 0.757 | 0.092 |
| TH | **0.989** | **1** | **0.993** |
| GP | 0.947 | **1** | 0.927 |
| GR | 0.97 | **1** | **0.993** |

Table IX combines consistent and erroneous alert results across both the increasing-load and steady-load scenarios to evaluate overall predictor accuracy. While autocorrelation and variance both performed poorly under the realistic network models, all predictors, except variance, were reasonably successful under the abstract model. As discussed elsewhere [6, 16], abstract network models lack essential, realistic traits (e.g., tiered forwarding speeds, finite buffers, and congestion-control protocols), and so congestion spreads outward from routers with high centrality. In realistic network models, congestion emerges at (low-centrality) edge routers and has difficulty spreading inward. For these reasons, abstract network models should not be used to evaluate congestion predictors.

Table IX also shows that under the realistic UDP model, threshold, growth-persistence, and growth-rate predictors all achieved perfect accuracy; however, Table V showed that the threshold predictor provided 3-4 minutes more warning time under all network models. While many short internet transactions use UDP procedures, longer flows are regulated by TCP congestion-control. Under the realistic TCP network model, the simplest predictor, threshold, provided highest accuracy: 98.9%. The residual 1.1% inaccuracy arises from five (fast) edge routers that alerted absent overload. (Recall, though, that queue lengths for those five routers would reach overload given more simulated time.) Among the five predictors investigated here, threshold seems best suited for further evaluation in real networks.

## VI. CONCLUSIONS AND FUTURE WORK

This paper evaluated five predictors that might be used to signal onset of congestion collapse. The paper also defined a method, and related measures, to evaluate predictor performance. In a series of simulation experiments, the paper compared predictors along four dimensions: implementation cost, accuracy, warning time, and reliability. The results, though obtained for a single network topology, apply generally because predictors run within individual routers and make decisions based on local state. Predictor performance does not depend directly on the size or structure of a network topology.

The experiment results support six, specific conclusions: (1) executing predictors incurs little router overhead; (2) simple predictors, e.g., threshold, give good accuracy, warning time, and persistence, especially under realistic network models; (3) two predictors, autocorrelation and variance, appear unsuitable due to many false alerts under steady load in realistic network models; (4) though including a buffer-capacity to queue-length relationship in the $P_{Threshold}$ condition reduces false alerts and could improve accuracy of autocorrelation and variance, but at the cost of decreased warning times; (5) autocorrelation and variance gave best average warning times; and (6) realistic network models should always be used to evaluate congestion predictors.

Future work remains to verify predictor results in emulated and real networks. Further, predictors include numerous parameters for which values must be selected; thus, sensitivity analysis is needed to determine the relative influence of parameters, and to select optimal parameter-value settings. An additional experiment is planned to investigate predictor performance under more complex traffic scenarios, e.g., cycles of increasing load followed by decreasing load, where the rate of increase and decrease varies with each cycle. Additional predictors will also be defined and evaluated.

## REFERENCES

[1] C. Albuquerque, B. Vickers, and T. Suda, "Network border patrol: preventing congestion collapse and promoting fairness in the Internet," IEEE/ACM Trans. on Networking, vol. 12 no. 1, 2004, pp. 173–186.

[2] D. Arrowsmith, R. Mondragón, J. Pitts, and M. Woolf, "Phase transitions in packet traffic on regular networks," ISSN 1103-467X, Institut Mittag-Leffler, 2004.

[3] R. Bush and D. Meyer, "Some Internet Architectural Guidelines and Philosophy," RFC 3439, 2002, 28 pages.

[4] B. Carreras, V. Lynch, I. Dobson, and D. Newman, "Critical points and transitions in an electric power transmission model for cascading failure blackouts," Chaos, 2007, vol.. 12 no. 4, pp. 985-994.

[5] C. Dabrowski, "The study of catastrophic event phenomena in communication networks," Comp. Sci. Rev., vol. 18, pp. 10−45.

[6] C. Dabrowski and K. Mills, "The Influence of Realism on Congestion in Network Simulations," NIST Technical Note 1905, 2016, 62 pages.

[7] V. Dakos, et al., "Slowing down as an early warning signal for abrupt climate change," Procs. of the National Academy of Sciences, 2008, vol. 105 no. 38, pp. 14308–14312.

[8] P. Echenique, J. Gómez-Gardenes, and Y. Moreño, "Dynamics of jamming transitions in complex networks," Europhys Lett, vol. 71 no. 2 2005, pp. 325-331.

[9] T. Hastie and R. Tibshirani, "Generalized additive models," Monographs on stats. and applied prob., Chapman & Hall, 1990, 352 pages.

[10] P. Hines, E. Cotilla-Sanchez, and S. Blumsack, "Topological models and critical slowing down: two approaches to power system blackout risk analysis," Procs. of the Hawaii Conf. on System Sciences, 2011, pp. 1–10.

[11] K. Iverson, "A programming language," Wiley & Sons, 1962, 315 pages.

[12] T. Karagiannis, M. Molle, and M. Faloutsos, "Long-range dependence: ten years of Internet traffic modeling," IEEE Internet Comp, vol. 8 no. 5, 2004, pp. 57–64.

[13] 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 Procs. of the Canadian Conf. on Electrical and Computer Engineering, 2007, pp. 360-363.

[14] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic," IEEE/ACM Trans. on Networking, vol. 2 no. 1, 1994, pp. 1–15.

[15] K. Lhaksmana, Y. Murakami, and T. Ishida, "Analysis of large-scale service network tolerance to cascading failure," IEEE Internet of Things Journal, vol. 3 no. 6, 2016, pp. 1159-1170.

[16] K. Mills and C. Dabrowski, "The need for realism when simulating network congestion," Procs. of the Spring Simulation Multi-Conf., 2016, pp. 228-235.

[17] K. Mills, E. Schwartz, and J. Yuan, "How to model a TCP/IP network using only 20 parameters," Procs. of Winter Simulation Conf., 2010, pp. 849-860.

[18] G. Mukherjee and S. Manna, "Phase transition in a directed traffic flow network," Phys Rev E, vol. 71 no. 6, 2005, 066108.

[19] Y. Rykalova, L. Levitin, and R. Brower, "Critical phenomena in discrete-time interconnection networks," Physica A, vol. 389, 2010, pp. 5259-5278.

[20] S. Sarkar, K. Mukherjee, A. Ray, A. Srivastav, and T. Wettergren, "Statistical mechanics-inspired modeling of heterogeneous packet transmission in communication networks," IEEE Trans on Syst, Man, and Cybernetics—Part B: Cybernetics, vol. 42 no. 4, 2012, pp. 1083-1094.

[21] M. Scheffer, et al., "Early warning signals for critical transitions," Nature, vol. 461 no. 3, 2009, pp. 53–59.

[22] R. Solé and S. Valverde, "Information transfer and phase transitions in a model of internet traffic," Physica A, vol. 289, 2001, pp. 595-605.

[23] W. Stallings, Queuing Analysis, 2000, 30 pages, see page numbered 4: http://www.box.net/shared/static/lu626umiib.pdf

[24] D. Stauffer and A. Aharony, "Introduction to percolation theory: revised second edition," Taylor & Francis, 1994, 192 pages.

[25] B. Tadić, G. Rodgers, and S. Thurner, "Transport on complex networks: flow, jamming and optimization," International Journal of Bifurcation and Chaos, vol. 17 no. 7, 2007, pp. 2363-2385

[26] D. Wang, N. Cai, Y. Jing, Y. and S. Zhang, "Phase transition in complex networks," Procs. Of American Control Conf., 2009, pp. 3310-3313.

[27] M. Woolf, D. Arrowsmith, R. Mondragón, and J. Pitts, "Optimization and phase transitions in a chaotic model of data traffic," Phys Rev E, vol. 66, 2002, 046106.

[28] xcorr, MATLAB v. 2015b, Mathworks, see:
https://www.mathworks.com/help/signal/ref/xcorr.html