Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

A LAYERED AND AGGREGATED QUEUING NETWORK SIMULATOR FOR DETECTION OF ABNORMALITIES

Junfei Xie

Department of Computing Sciences
Texas A&M University-Corpus Christi
6400 Ocean Dr, Corpus Christi, TX, USA

Chenyuan He
Yan Wan

Department of Electrical Engineering
University of Texas at Arlington
416 Yates Street, Arlington, TX, USA

Kevin Mills
Christopher Dabrowski

National Institute of Standards and Technology
Gaithersburg, MD, USA

ABSTRACT

Driven by the needs to monitor, detect, and prevent catastrophic failures for complex information systems in real-time, we develop in this paper a discrete-time queuing network simulator. The dynamic model for the simulator abstracts network dynamics by taking an aggregated and layered structure. Comparative studies verify capabilities of the simulator in terms of accuracy and computational efficiency. We illustrate the model structure, flow processing mechanisms, and simulator implementation. We also illustrate the use of this simulator to detect distributed denial-of-service (DDoS) flooding attacks, based on a cross-correlation-based measure. Finally, we show that the layered structure provides new insights on the spatiotemporal spread patterns of cascading failure, by revealing spreads both horizontally within a sub-network and vertically across sub-networks.

1 INTRODUCTION

Malicious attacks or router failures may lead to catastrophic performance degradation in complex information systems (e.g. the Internet, computing grids, and information clouds) (Dabrowski and Hunt 2011, Yuan and Mills 2005b). The prevention of network failures requires a computationally effective simulator that can monitor (or measure) system dynamics, detect abnormal events, and predict catastrophic performance degradation, all in real time. This can be a challenging task for large-scale networks, considering uncertain varying traffic demands, complicated data flow interactions, and unexpected resource reductions. The purpose of this paper is to develop a network simulator that is realistic enough to capture complicated flow dynamics and interactions, and is also simple and abstract enough to facilitate fast detection of network abnormalities, as a step toward the prediction and prevention of network failures.

In existing network studies for complex information systems, packet-oriented discrete-event simulators (DES) have typically been used (Yuan and Mills 2006, Garetto et al. 2001). These simulators can truthfully capture network behaviors, as they simulate the processing of each data packet triggered by events such as the arrivals of packets, time-outs and receipts of acknowledgment signals, using if-then-types of operations on state automata. These simulators are typically not used for real-time prediction and prevention studies

978-1-5386-3428-8/17/$31.00 ©2017 IEEE

for two reasons: i) the computational load for simulating packet-level traffic can be large for networks of high packet volume, ii) detailed knowledge of packet-level traffic for future planning is not necessarily known.

Continuous-time and fluid-based DES (Nicol and Yan 2004, Gu et al. 2004) have been investigated to address the computational issue of packet-oriented DES. They track the rate changes of packet flows instead of the processing of individual packets. These discrete-event models suffer from the "ripple effects" with growth of network size and complexity (Liu et al. 1999). To address this issue, discrete-time fluid models have then been exploited (Yan and Gong 1999, Yan 1998, Zheng et al. 2007). Distinct from DES which are triggered by events, discrete-time fluid models are triggered by clock clicks, and abstract both packet flows and processing times. These models are more suitable for time-critical decision-making applications, due to the reduced-order representation and theoretical queuing system analyses which can be brought to reduce evaluation and simulation time (Misra et al. 1999, Misra et al. 2000). However, existing discrete-time fluid models typically consider simplified network topologies and flow processing schemes, and are not developed for real-time detection and autonomous prevention of network abnormalities. Also related, hybrid models, which equip time-driven dynamics with jump conditions, bring extra flexibility to capture complicated packet processing protocols (Yi and Shakkottai 2007, Lee et al. 2007).

In this paper, we develop a network simulator based on an aggregated discrete-time fluid-based queuing network model to facilitate quick detection of network abnormalities, as a step toward on-line failure prediction and prevention. The network model adopts a layered network structure, each layer of which constitutes a sub-network of an aggregated source-destination (S-D) pair. Packets are aggregated into flows, and packet processing procedures (such as rerouting and packet drops) at a flow level are triggered at aggregated time instances. Uncertain demands, attacks, and router failures can all be naturally captured. This aggregated abstraction of packets, S-D pairs, and time resolutions, improves the computational efficiency for real-time monitoring, detection, prediction, and prevention tasks, and makes the network model scale well with demand volume and network size. Simulators are prototyped using Matlab and then implemented using SLX (Henriksen 2000). To understand performance of this queuing network model and simulator, we conduct a series of comparative studies with a packet-oriented DES baseline model. Simulation studies demonstrate promising performance in terms of accuracy and efficiency, especially for networks of large size and heavy demand volumes. These comparative studies also reveal the trade-off between accuracy and efficiency, which help users to choose appropriate aggregation resolutions for their studies.

We also study using the layered discrete-time aggregated queuing network simulator to detect abnormalities including both distributed denial-of-service (DDoS) flooding attacks and cascading failures as examples. DDoS flooding attack is a malicious attack difficult to detect due to the distributed sources where attacks are initiated. DDoS attacks quickly consume resources and induce overload at the target under attack. We show in this paper that our simulator can detect DDoS attacks using the cross-correlation-based method (Yuan and Mills 2005b), despite the aggregation and abstraction adopted in our modeling framework. A variety of common DDoS attack behaviors are also studied.

Another major concern for complex information systems is cascading failure, which is caused by complicated nonlinear network interactions (Coffman et al. 2002, Liao et al. 2004). Such failures may initiate locally, and trigger a series of flow re-distributions that cause many routers to fail. In this paper, we use our layered and aggregated queuing network simulator to characterize cascading failures. The layered structure allows the structural analyses of flow re-distribution patterns horizontally within a layer and also vertically across layers. Two failure scenarios, including a single router failure and excessive demands, are studied to demonstrate capabilities of the simulator.

In the remainder of this paper, we first describe the layered and aggregated queuing network model and its simulator implementation in Section 2. We then analyze the accuracy and efficiency of the simulator in Section 3, through a comparative study with a packet-oriented DES. In Section 4, we investigate the feasibility of using our simulator and the cross-correlation-based method to detect DDoS attacks. In

Section 5, we study the capability of our simulator in capturing and analyzing cascading failures. Section 6 concludes the paper.

2 LAYERED AND AGGREGATED DISCRETE-TIME QUEUING NETWORK MODEL

The queuing network model has several features that make it suitable for real-time detection of abnormalities, as a step toward the prediction and prevention of network failures. First, the aggregated model is a *reduced order* representation of the original network dynamics, and thus consumes less computational time for real-time operations. Second, the queuing network representation captures complicated *spatiotemporal* network interactions and *transient impacts* of attacks and resource reductions. This feature permits the implementation and evaluation of failure prevention solutions. Third, the layered model *tracks the sources and destinations of flows*. It also captures the *spatiotemporal spread of network performance degradations* within a S-D sub-network and across sub-networks to permit early detection of network failures. Fourth, *uncertain* demand models can be easily integrated into the model to predict and evaluate system performance under uncertain future demands. In this section, we describe the model from several aspects: network structure, flow rerouting, and flow processing at routing nodes.

2.1 Network Structure

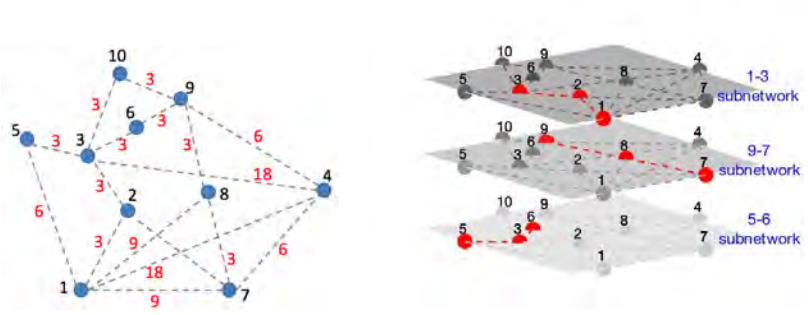
The network is composed of three types of nodes: i) source nodes (denoted as $s \in S$, where S is the set of source nodes) with data flows injected from the outside of the network, ii) destination nodes (denoted as $d \in D$, where D is the set of destination nodes) with flows leaving the network, and iii) routing nodes (denoted as $i \in \mathscr{R}$, where $\mathscr{R}$ is the set of routing nodes) which forward received data flows to neighboring nodes according to specified routing rules (shortest path chosen in this study, but can be others). Without loss of generality, we allow some routing nodes to also serve as source or destination nodes, i.e., $S \subseteq \mathscr{R}$ and $D \subseteq \mathscr{R}$. is defined on a graph Γ. In particular, nodes i and j are considered as neighbors if a one-hop link between i and j exists. The propagation delay from node i to j is described by a_{ij}. If the one-hop link from i to j does not exist, a_{ij} is assigned ∞. The matrix $A \in R^{n\times n}$ captures network topology defined on graph Γ, where n is the total number of nodes. a_{ij} is the element at the ith row and jth column. We note that the matrix A is updated on the occurrence of node failures to reflect the dynamic changes of network topologies.

We emphasize that information systems are different from many other dynamical systems of interacting network elements (e.g., power networks and epidemic spread networks) in that each departing flow has a pre-determined destination. The model must be able to track flow destinations to facilitate failure prevention strategies such as rerouting. Moreover, the capability of source tracking helps detection and prevention of abnormalities such as DDoS attacks. To facilitate the tracking of source and destination, we decompose the network into multiple layers indexed by S-D pairs (see Figure 1 for an example and also a related development in (Wan et al. 2013) for a different application). We use $\mathscr{L}$ to denote the set of all valid S-D sub-networks. The multiple sub-networks are stacked together to form the complete network.

2.2 Flow Rerouting

Within each S-D sub-network, we assume that a flow chooses the shortest path to the destination. Many metrics can be used to determine the shortest path, and in our study the shortest path is defined on the smallest total propagation delays, and is selected based on the matrix A according to the Floyd algorithm (Floyd 1962). The routing information is saved in the *routing table* $\mathscr{T}$, with the element $\mathscr{T}_{ij}$ in the ith row and jth column representing the next hop of node i along the path to the destination node j.

When a node fails, flows are re-distributed, through updating the matrix A and routing table $\mathscr{T}$. For flows that have already entered a disconnected link (i, j) with either node on the ends fails, they will continue to enter node j. If node j is functioning, the flow will be transmitted; otherwise the flow will stay in the queue of node j. In case when packet dropping is considered, all packets in the queue of node



(a) (b)

Figure 1: Illustration of the layered structure of the queuing network model, using a small-scale network of 10 nodes and 3 S-D sub-networks. a) The network structure, where red numbers on links represent the propagation delays a_{ij} between two nodes. b) 3 network layers indexed by S-D pairs, where nodes and links belonging to each sub-network are marked in red.

j will be dropped. In this study, we disable the packet dropping feature with the purpose to investigate the capability of our model in capturing cascading failures of routers. Failed nodes will be restored after certain waiting time T_r (Wang et al. 2008).

2.3 Flow Processing at Routing Nodes

The layered S-D sub-networks interact with each other at routing nodes. In particular, a routing node i processes aggregated flows from all S-D sub-networks with its limited processing capabilities, denoted as C_i volume of packets. Under the simple assumption that routing nodes with more connections have higher processing capacities (Wang et al. 2008), we define C_i to be proportional to its node degree, i.e., $C_i = \alpha \cdot h_i$, where h_i is the degree of node i, and α is a scalar.

Routing nodes fairly process flows in multiple S-D sub-networks. Packets that exceed the processing capability of each routing node are accumulated in the queue. The total queue length (i.e., the volume of packets accumulated) at node i is called the total backlog and denoted as b_i. If b_i exceeds its buffer size L_i, the routing node fails. Here L_i is assumed to be $C_i + c$, where c is a constant. Other ways of defining L_i can be found in the literature. For instance, $L_i = c'N_i$ reflects that the buffer size of a routing node is proportional to its processing capacity (Wang et al. 2008), where c' is a constant.

The flow processing and propagation dynamics in a S-D sub-network are mathematically described as follows. At each time step k and each routing node in a S-D sub-network, the **inflow** (volume of packets arriving at the routing node per time interval) $u_{sdi}[k]$, **backlog** $b_{sdi}[k]$, and **outflow** (volume of packets leaving the routing node per time interval) $f_{sdi}[k]$ are tracked.

1) *The update of inflows* $u_{sdi}[k+1]$. The inflow, $u_{sdi}[k]$, is the accumulated outflow propagated from all nodes one-hop prior to node i in the sub-network sd. If i is the source node, the inflow is 0.

$$u_{sdi}[k+1] = \begin{cases} 0, & if\ i = s; \\ \sum_{\forall a_{ji} \neq \infty} f_{sdj}[k - a_{ji}], & else. \end{cases} \quad (1)$$

2) *The update of outflows* $f_{sdi}[k+1]$ *and backlogs* $b_{sdi}[k+1]$. The update of outflows and backlogs at the routing nodes follows the first-in-first-out rule and the fairness principle for multiple arriving flows. Denote the total inflow to node i as $u_i[k] = \sum_{\forall (s,d) \in \mathscr{L}} u_{sdi}[k]$, and the total backlog at node i as $b_i[k] = \sum_{\forall (s,d) \in \mathscr{L}} b_{sdi}[k]$. The following cases are considered when node i does not fail. If $b_i[k] \leq C_i - u_i[k+1]$, all current backlogs and new arriving flows can be processed. If $C_i - u_i[k+1] < b_i[k] \leq C_i$, all current backlogs can be processed, and new arriving flows in each sub-network at time $k+1$ are processed proportional to the fraction of arriving flows in each sub-network. If $b_i[k] > C_i$, only a portion of current backlogs in each sub-network

can be processed proportional to the relative fraction. Remaining backlogs and the new arrival flows will form the new backlogs at time $k+1$.

$$\begin{aligned} f_{sdi}[k+1] &= \begin{cases} b_{sdi}[k]+u_{sdi}[k+1], & if\ b_i[k] \le C_i - u_i[k+1]; \\ b_{sdi}[k], & if\ C_i - u_i[k+1] < b_i[k] \le C_i\ and\ u_i[k+1] = 0; \\ \frac{u_{sdi}[k+1]}{u_i[k+1]}(C_i - b_i[k]) + b_{sdi}[k], & if\ C_i - u_i[k+1] < b_i[k] \le C_i\ and\ u_i[k+1] \neq 0; \\ \frac{b_{sdi}[k]}{b_i[k]} C_i, & else; \end{cases} \\ b_{sdi}[k+1] &= b_{sdi}[k] + u_{sdi}[k+1] - f_{sdi}[k+1]. \end{aligned} \tag{2}$$

In cases when node i fails, no flows are allowed to pass, i.e., $f_{sdi}[k+1] = 0$. If packet dropping is considered, we also set $b_{sdi}[k+1] = 0$.

At each source, the backlog is 0, and the outflow is the total flow injected from this source to the destination, denoted as $u_{sd}[k]$. At each destination, arriving packets are accumulated in the destination queue of infinite buffer size. Specifically,

$$\begin{aligned} f_{sdi}[k+1] &= \begin{cases} u_{sd}[k+1], & if\ i = s; \\ 0, & if\ i = d; \end{cases} \\ b_{sdi}[k+1] &= \begin{cases} 0, & if\ i = s; \\ b_{sdi}[k] + u_{sdi}[k+1], & if\ i = d. \end{cases} \end{aligned} \tag{3}$$

2.4 Simulator Implementation

We prototyped the layered and aggregated discrete-time queuing network model using Matlab first and then built the simulator using SLX for computational efficiency (Henriksen 2000). Both simulators require two input files, network topology matrix A and the demand file which stores data demands injected to each S-D sub-network at each time step k, i.e., $u_{sd}[k]$. The simulation terminates either when a predefined timeout occurs or when no path exists between any S-D pair. The output file, storing the state $\mathscr{S}_{sdi}[k] = [u_{sdi}[k]\ f_{sdi}[k]\ b_{sdi}[k]]^T$ of each node i in each sub-network sd at each time step k, is generated for visualization and analysis, where T denotes the transposition.

2.4.1 Matlab-based Simulator

In the Matlab-based simulator, the processing of flows is achieved through storing and updating the state table $\mathscr{S} = \{\mathscr{S}_{sdi}[k]\}$ according to Equations 1-3. Active flow routes are maintained by a link status table G_{sd}, which stores the active link information for each sub-network sd. In particular, each element in G_{sd} is a three-tuple (i, j, f), where i is the current node, j is the next hop, and f indicates the status of the link and nodes at the ends. Specifically, $f = 0$ means that the link (i, j) belongs to the shortest path from s to d; $f = 1$ indicates that both nodes at the ends of the link (i, j) are functioning, however the link does not belong to the shortest path from s to d; and $f = 2$ denotes that either $i \in F$ or $j \in F$, where F is the set of failed nodes. When a routing node i fails, set F, matrix A and the link status table G_{sd} are updated to capture the rerouting of flows. When updating G_{sd}, the f entries for existing links are updated, and new links along alternative rerouted paths are also added to G_{sd} with $f = 1$.

2.4.2 SLX-based Simulator

The SLX-based simulator improves computational efficiency by leveraging several features of SLX, such as objective-oriented design, capability for expressing parallelism, and the ability to suspend and resume system states (Henriksen 2000). We define four classes of objects, including source, destination, routing node and data flow, each of which has associated attributes and/or operations. For instance, each flow is tagged with attributes including source, destination, creation time, flow size, current node it resides and

the next hop determined by the routing table $\mathscr{T}$. The processing of flows at routing nodes is achieved by maintaining three local queues (i.e., inflow, outflow and backlog) at time instance-triggered events. Each flow exported by a node will arrive at its next hop after a propagation delay determined by matrix A. The SLX-based simulator outperforms the Matlab-based simulator in computational efficiency. In particular, the object-oriented design avoids the use of cumbersome tables $\mathscr{S}$ and G_{sd} to track flows with expensive computation. The intra-object parallelism feature of SLX simplifies the description of parallel operations of complex objects. A comparison study between the two simulators is provided in Section 3.2.

3 COMPARATIVE SIMULATION STUDIES

In this section, we investigate performance of the SLX-based queuing network simulator through comparative simulation studies. We first evaluate the trade-off between error rate and computational time with different aggregation levels along the time scale. We then evaluate the computational efficiency of the SLX-based queuing network simulator with the increase of demand volume and network size.

3.1 Impact Analysis of Aggregation on Error Rate and Computational Efficiency

As packet-oriented DES is considered as truthful for complex information systems, we here build such a model using SLX as the benchmark to evaluate the error rate of our SLX-based discrete-time queuing network simulator.

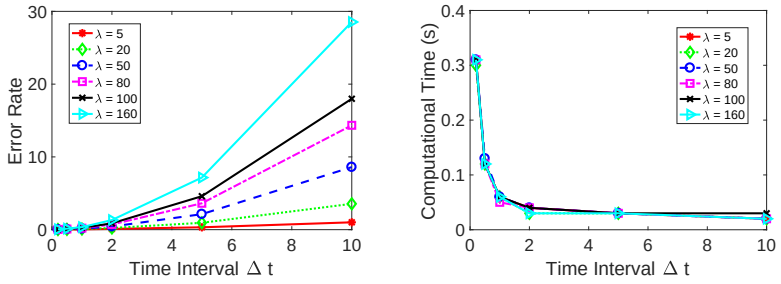
The major difference between our queuing network simulator and packet-oriented DES is the aggregation of packets and processing times. In particular, packets generated within each time interval are aggregated into flows at source nodes, which are then processed at routing nodes at each time step. We here use the 10-node network shown in Figure 1(a) to study the impact of an aggregation indicator, time interval Δt, on the performance in terms of accuracy and computational efficiency. Each node in the network functions as source, destination, and router. A total of 90 S-D sub-networks are considered. Flows injected into the network from each source are generated from a Poisson distribution with mean λ, which controls the demand volume in the network. Node capacity and buffer size parameters are set to large values to capture normal flows without node failures. Figures 2(a)-2(b) show the accuracy and efficiency of our simulator under different time intervals and demand volumes. Accuracy is measured using *error rate* defined as

$$error\ rate = \frac{\sqrt{\sum_{i=1}^{n}\sum_{k=1}^{N}(P_{si}[k]-P'_{si}[k])^2+(P_{ri}[k]-P'_{ri}[k])^2+(P_{di}[k]-P'_{di}[k])^2}}{nN}, \tag{4}$$

where $P_{si}[k]$ and $P'_{si}[k]$ represent the volume of packets sent within the k-th time interval by node i which functions as a source node, using our SLX-based queuing network simulator and packet-oriented DES, respectively. Similarly, $P_{ri}[k]$ and $P'_{ri}[k]$ are for data sent by routing node i; and $P_{di}[k]$ and $P'_{di}[k]$ are for data received by node i which functions as a destination node. N is the total number of time steps, 500 in this study. Figures 2(a)-2(b) show that the increase of time interval leads to lower accuracy (larger error rate) but higher efficiency (smaller computational time). The selection of time interval needs to balance between accuracy and efficiency. $\Delta t = 1$ is selected for the rest of our simulation studies. Another observation from the two figures is that the increase of Poisson mean λ decreases accuracy, but does not impact computational efficiency much due to the aggregated processing of all packets in the time interval as a whole, which indicates the scalability of our model to demand volume.

3.2 Analysis of Computational Efficiency

We investigate the computational efficiency of our model with increase of demand volume and network size. In the first experiment, we study the impact of demand volume on computational efficiency for the 10-node network (see Figure 1(a)), by varying the Poisson mean λ. Figure 3(a) indicates that our model (implemented using SLX) is more efficient than the packet-oriented DES, and the SLX-based simulator is more efficient than the Matlab-based simulator. Another observation is that while the computational

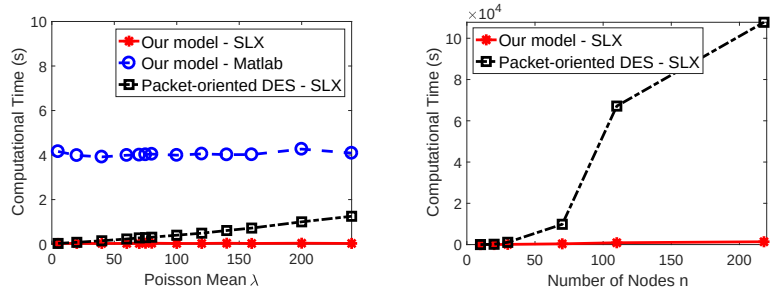


(a) (b)

Figure 2: a) Accuracy and b) efficiency of our model under various time intervals and data demands.

time required by packet-oriented DES increases linearly with increase of λ, the computational time for our model (implemented using both Matlab and SLX) remains almost constant, as is also shown in Figure 2(b). This further illustrates the nice scalability of our model to demand volume.

In the second experiment, we study the impact of network size on efficiency, by varying the number of nodes n. In particular, we apply the Erdős Rényi (ER) model (Bollobás 1998) to generate random networks of different sizes. We also fix the total demand volume injected into each network. Figure 3(b) shows the comparison results for a particular demand volume. Clearly, the computational time for the packet-oriented DES increases much faster than our model with growth in network size, indicating better scalability of our model to network size compared to packet-oriented models.



(a) (b)

Figure 3: Comparison of our model with the packet-oriented DES model in terms of computational efficiency under various a) demand volumes with fixed network size, and b) network sizes with fixed total demand.

4 CROSS-CORRELATION-BASED METHOD TO DETECT DDoS ATTACKS

DDoS flooding attack is difficult to detect, as attack packets originate from a large number of attack sources, each of which is disguised to behave normally. The widely applied congestion control strategy, TCP, is incapable of detecting DDoS attacks (Yuan and Mills 2005b). In this section, we show that the SLX-based discrete-time queuing network simulator can detect DDoS attacks using the cross-correlation-based method introduced in (Yuan and Mills 2005b, Yuan and Mills 2005a). Despite the aggregation of packets and processing times and hence the reduced computational load, the method can still analyze collective effects of network traffic and detect DDoS attacks.

4.1 Cross-Correlation-based Method using the Random Matrix Theory

The cross-correlation-based method was developed based on the random matrix theory (Barthélemy et al. 2002), which defines a weight vector to quantify spatial-temporal correlations among different routing domains. DDoS flooding attacks can be detected as they cause higher spatial-temporal correlations. The procedures to calculate the weight vector are briefly summarized as follows.

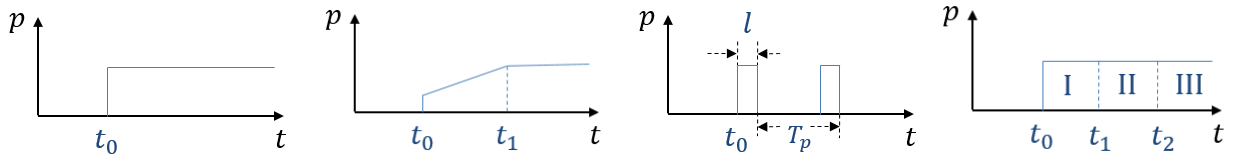
Step 1: Calculate the cross-correlation matrix. For a network of n routing nodes, n_o ($n_o < n$) nodes are selected to measure flow information. Let x_{ij} denotes the flow volume coming from node i to node j. The cross-correlation matrix, $\mathscr{C} \in R^{K\times K}$, $K = n_o n$, can be calculated by $\mathscr{C}_{(ij)(kl)} = < \tilde{x}_{ij}\tilde{x}_{kl} >_{t_w}$, where $\tilde{x}_{ij} = \frac{x_{ij}-\bar{x}_{ij}}{\sigma_{ij}}$. $\bar{x}_{ij}$ and σ_{ij} are the mean and variance of x_{ij}. $< \cdot >_{t_w}$ is a mean operator over a measurement interval t_w.

Step 2: Find the eigenvector corresponding to the largest eigenvalue of the cross-correlation matrix. This eigenvector, denoted as $w^{max} \in R^K$, can be computed by $\mathscr{C}w = \lambda_w w$, where λ_w is the largest eigenvalue.

Step 3: Calculate the weight vector. Decompose eigenvector w^{max} into n sub-vectors, i.e., $w^{max} = [w_1^{max}, w_2^{max}, ..., w_n^{max}]^T$, where $w_i^{max} \in R^{n_o}$ corresponds to the ith routing node. The jth element of sub-vector w_i^{max}, denoted as w_{ij}^{max} ($i \le n$ and $j \le n_o$), indicates the contribution of the jth measurement node to the ith routing node. The weight W_i for each routing node i is then computed by $W_i = \sum_{j=1}^{n_o} (w_{ij}^{max})^2$, which informs the contribution of flows from all measurement nodes to the ith routing node. The weight vector $W = [W_1, W_2, ..., W_n]^T$ is used to draw a spatial-temporal map for detecting DDoS attacks.

4.2 Early Detection of DDoS Attacks

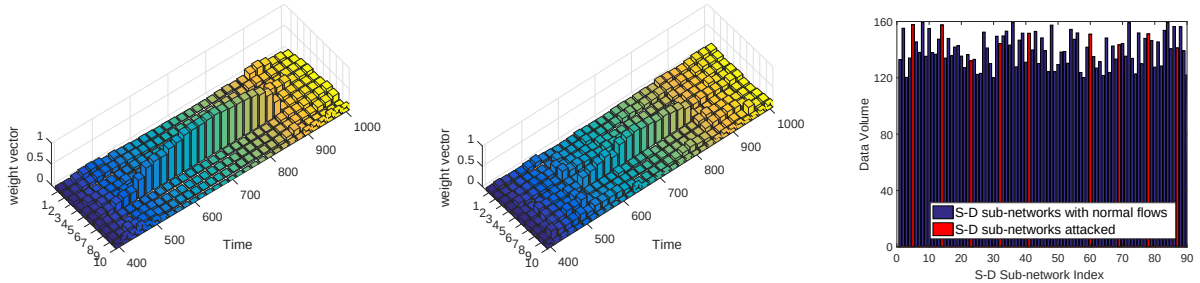
We use the 10-node network of 90 S-D pairs shown in Figure 1(a) as an illustrative example to conduct this cross-correlation analysis. Larger-scale networks have also been investigated, which lead to similar results and thus are eliminated here due to the page limit. The flows injected into the network from each node follow a uniform distribution $U(120, 160)$. $\alpha = 2000$ and $c = 60$ so that no nodes fail. Four nodes (3, 5, 8, 9) are randomly picked as the measurement nodes. $N = 1000$, $\Delta t = 1$ and $\tau_w = 400$. The time series of W are obtained by moving the time window τ_w ahead every 20 time units. We now consider four typical DDoS bandwidth attack modes: constant-rate, increasing-rate, pulsing, and subgroup (see (Yuan and Mills 2005b) and Figure 4 for an illustration). Node 6 is selected as the target under attack. In each sub-network with destination node 6, an extra small amount of attack packets are generated, denoted as p volume of packets per time interval. Let us next describe each attack mode and show the corresponding detection results.



(a) (b) (c) (d)

Figure 4: DDoS flooding attack modes: a) constant-rate, b) increasing-rate, c) pulsing, and d) subgroup.

1) Constant-Rate Attack Each attack source generates a constant volume of attack packets over time. Figure 5(a) shows the spatial-temporal map of the attack starting at $t_0 = 500$ with $p = 10$. Similar to the results obtained in (Yuan and Mills 2005b), the target under attack (node 6) and start time of attack can be detected. Of interest, DDoS attacks cannot be detected by directly measuring the demand volume injected into the network as indicated in Figure 5(c). Figure 5(b) shows a weaker spatial-temporal map with p reduced to 5, indicating limited capability of the cross-correlation-based method in detecting weak attacks.

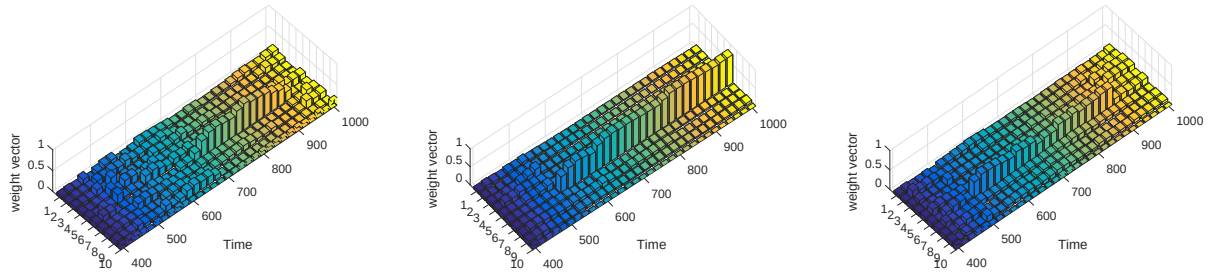


(a) (b) (c)

Figure 5: Spatial-temporal maps of constant-rate attack with a) $p = 10$ and b) $p = 5$. c) Data volume injected into each S-D sub-network at time $t = 800$ with attacked S-D sub-networks highlighted in red.

2) Increasing-Rate Attack The increasing-rate attack consumes network resources gradually to delay the time to be detected. In this study, we initiate increasing-rate attacks at $t_0 = 500$ with $p = 1$, and gradually increase p until it reaches 10 at $t_1 = 80$. The attack rate p then keeps constant. The spatial-temporal map shown in Figure 6(a) detects the victim at around $t = 700$.

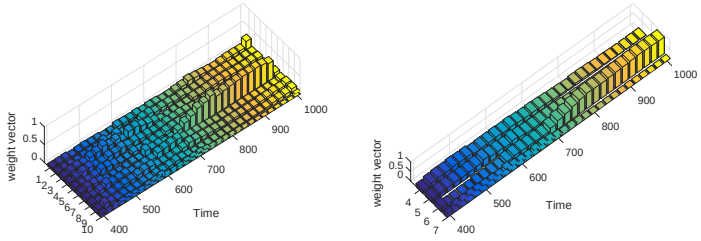
3) Pulsing Attack Periodic attacks are launched to avoid detection, which however can still be detected by the spatial-temporal map shown in Figure 6(b), where $t_0 = 500$, $l = 60$, $T_p = 300$ and $p = 10$. The TCP-targeted attack is a special case of the pulsing attack with shorter period and higher attack rate, so as to suppress TCP flows by locking them in the time-out states. Such DDoS attack can also be detected using the cross-correlation analysis as shown in Figure 6(c), where $l = 1$, $T_p = 2$, and $p = 80$.



(a) (b) (c)

Figure 6: Spatial-temporal maps of a) increasing-rate attack with p increasing from 1 to 10 from $t_0 = 500$ to $t_1 = 800$, b) pulsing attack with $l = 60$, $T_p = 300$, and $p = 10$, and c) TCP-targeted attack with $l = 1$, $T_p = 2$, and $p = 80$.

4) Subgroup Attack The subgroup attack initiates attacks at different groups of attack sources during different time periods, so that other groups can resume the attack if one of the groups is detected. Similar to the setup in (Yuan and Mills 2005b), we divide attack sources into 3 groups. The first group launches attacks at $t_0 = 500$. The second group continues attacks from $t_1 = 680$ to $t_2 = 860$, from which moment the third group starts to make attacks. The attack rate p is set to 15 for all three groups. Note that the total volume of attacks generated in this case is only half of that for the constant-rate attack. This explains the weak visibility of the attack in the spatial-temporal map (Figure 7(a)) at the beginning of the attack. Similar patterns can also be captured by measuring at fewer nodes (see Figure 7(b)).



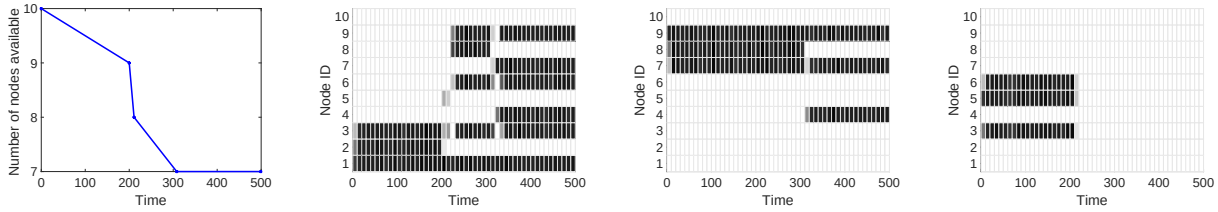
(a) (b)

Figure 7: Spatial-temporal maps of subgroup attack measured at a) all 10 nodes, and b) nodes 4, 5, 6, 7.

5 UNDERSTANDING AND VISUALIZING CASCADING FAILURES

The layered network model provides additional information to capture cascading failures. We continue to use the 10-node network as the example. For better illustration, we consider three S-D sub-networks as illustrated in Figure 1(b). Flows injected into each sub-network are generated from a Poisson distribution with mean of 50, which is typically not sufficient to trigger node failures. $\alpha = 20$, $c = 60$, and the waiting time for a failed node to recover is set to $T_r = \infty$. We consider the following two scenarios.

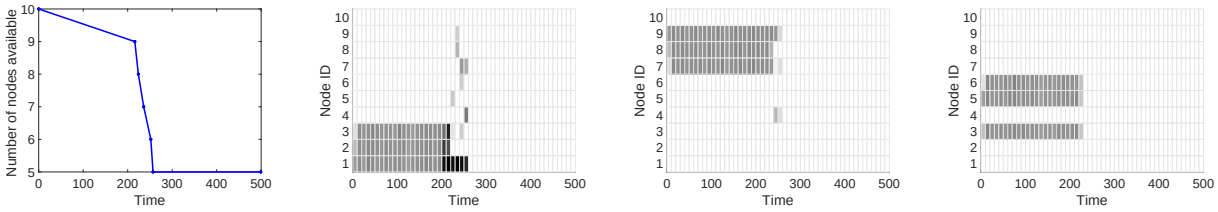
1) Scenario 1: single node failure. We manually fail node 2 at $t = 200$. As shown in Figure 8(a), another two nodes fail at time $t = 211$ and $t = 308$, respectively. To find out the cause of these additional two node failures, we measure the inflows to each node in all three S-D sub-networks. For better demonstration, we aggregate inflows at a time interval of 10 time units, normalize them to the range of $(0,1)$ with 1 corresponding to the largest inflow over all sub-networks, and visualize the results with darker colors indicating higher inflows (see Figures 8(b)-8(d)). Figure 8(b) shows that the failure of node 2 causes flows in sub-network 1-3 to be rerouted to node 5. These rerouted flows compete for resources with flows in the sub-network 5-6 (see Figure 8(c)), leading to failure of node 5 at time $t = 211$, as indicated in Figures 8(b) and 8(d). Upon the failure of node 5, flows in sub-network 1-3 are then rerouted to an alternative path consisting of nodes 6, 8, and 9, as indicated in Figure 8(b). These flows increase the burden of node 8, which also processes flows in the 9-7 sub-network, and finally causes this node to fail at time $t = 308$. The failure of node 8 then triggers another round of flow re-distributions, as shown in Figures 8(b)-8(c).



(a) (b) (c) (d)

Figure 8: a) Number of functioning nodes versus time after one router is failed manually. Visualization of inflows to each node in the sub-networks b) 1-3, c) 9-7, and d) 5-6.

2) Scenario 2: excessive demand. We increase the mean of the Poisson flow injected to the sub-network 1-3 to 110 from $t = 200$. Figure 9(a) shows that the first node failure occurs at $t = 216$. Within an additional 41 time units, another 4 nodes fail, which causes the breakdown of the whole network. Figure 9(b) shows that the increase of demands in the 1-3 sub-network starts from $t = 200$, which first causes node 2 to fail. The failure of node 2 then leads to a series of flow re-distributions and the spread of node failures to sub-networks 9-7 and 5-6.



(a) (b) (c) (d)

Figure 9: a) Number of functioning nodes versus time after an introduction of excessive demand. Visualization of inflows to each node in the sub-networks b) 1-3, c) 9-7, and d) 5-6.

6 CONCLUSIONS

We develop in this paper a layered and aggregated discrete-time queuing network simulator to detect abnormalities, as a step toward real-time failure prediction and prevention. Comprehensive comparative studies are conducted to demonstrate the realism and efficiency of the simulator. This simulator allows an effective detection of DDoS attacks of various types, using a cross-correlation-based method. The spread of cascading failures within and across sub-networks can also be easily captured and visualized using the layered network representation. In the future, we will model TCP protocols and investigate behaviors of TCP traffics with packet dropouts under attacks and node failures.

ACKNOWLEDGMENTS

We would like to thank National Institute of Standards and Technology (NIST), and National Science Foundation for the support under grant CPS-1714826.

REFERENCES

Barthélemy, M., B. Gondran, and E. Guichard. 2002. "Large scale cross-correlations in Internet traffic". *Physical Review E* 66 (5): 056110.

Bollobás, B. 1998. "Random graphs". In *Modern Graph Theory*, edited by S. Axler, F. Gehring, and K. Ribet, 215–252. New York: Springer Science & Business Media, Inc.

Coffman, E. G., Z. Ge, and V. Misra. 2002. "Network resilience: exploring cascading failures within BGP". In *Proceedings of 40th Annual Allerton Conference on Communications, Computing and Control*. Monticello.

Dabrowski, C., and F. Hunt. 2011. "Identifying Failure Scenarios in Complex Systems by Perturbing Markov Chain Models". In *Proceedings of the 2011 Pressure Vessels and Piping Division Conference*, 17–21.

Floyd, R. W. 1962. "Algorithm 97: shortest path". *Communications of the ACM* 5 (6): 345.

Garetto, M., R. L. Cigno, M. Meo, and M. A. Marsan. 2001. "A detailed and accurate closed queueing network model of many interacting TCP flows". In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Volume 3, 1706–1715.

Gu, Y., Y. Liu, and D. Towsley. 2004. "On integrating fluid models with packet simulation". In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, Volume 4, 2856–2866.

Henriksen, J. O. 2000. "SLX: the X is for extensibility". In *Proceedings of 2000 Winter Simulation Conference*, edited by J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 183–190. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Lee, J., S. Bohacek, J. P. Hespanha, and K. Obraczka. 2007, June. "Modeling communication networks with hybrid systems". *IEEE Transactions on Networking* 15 (3): 630–643.

Liao, H., J. Apt, and S. Talukdar. 2004, December. "Phase transitions in the probability of cascading failures". In *Proceedings of Electricity Transmission in Deregulated Markets, Conference at Carnegie Mellon University*. Pittsburgh PA.

Liu, B., Y. Guo, J. Kurose, D. Towsley, and W. Gong. 1999, June 28–July 1. "Fluid simulation of large scale networks: issues and tradeoffs". In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, Volume IV, 2136–3142. Las Vegas, NV.

Misra, V., W. B. Gong, and D. Towsley. 1999. "Stochastic differential equation modeling and analysis of TCP window size behavior". In *Proceedings of PERFORMANCE'99*, Volume IV. Istanbul, Turkey.

Misra, V., W. B. Gong, and D. Towsley. 2000. "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED". *ACM SIGCOMM Computer Communication Review* 30 (4): 151–160.

Nicol, D. M., and G. Yan. 2004. "Discrete event fluid modeling of background TCP traffic". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 14 (3): 211–250.

Wan, Y., C. Taylor, S. Roy, C. Wanke, and Y. Zhou. 2013, September. "Dynamical queuing network model for flow contingency management". *IEEE Transactions on Intelligent Transportation Systems* 14 (6).

Wang, J., Y. Liu, X. Sun, and Y. Jiao. 2008. "Modeling cascading failures in congested internet". In *Proceedings of the 9th International Conference for Young Computer Scientists*, 1499–1504.

Yan, A. 1998. *On some modeling issues in high speed networks*. Ph. D. thesis, University of Massachusetts Amherst, Amherst, MA.

Yan, A., and W.-B. Gong. 1999. "Time-driven fluid simulation for high-speed networks". *IEEE Transactions on Information Theory* 45 (5): 1588–1599.

Yi, Y., and S. Shakkottai. 2007. "FluNet: a hybrid internet simulator for fast queue regimes". *Computer Networks* 51 (18): 4919–4937.

Yuan, J., and K. Mills. 2005a. "Macroscopic dynamics in large-scale data networks". In *Complex Dynamics in Communication Networks*, edited by G. V. Ljupco Kocarev, 191–211. Berlin: Springer Berlin Heidelberg.

Yuan, J., and K. Mills. 2005b, October-December. "Monitoring the macroscopic effect of DDoS flooding attacks". *IEEE Transactions on Dependable and Secure Computing* 2 (4): 1–12.

Yuan, J., and K. Mills. 2006, May. "Simulating timescale dynamics of network traffic using homogeneous modeling". *Journal of Research of the National Institute of Standards and Technology* 111 (3): 227–242.

Zheng, J.-F., Z.-Y. Gao, and X.-M. Zhao. 2007. "Modeling cascading failures in congested complex networks". *Physica A: Statistical Mechanics and its Applications* 385 (2): 700–706.

AUTHOR BIOGRAPHIES

JUNFEI XIE is an Assistant Professor from Texas A&M University-Corpus Christi. She holds a Ph.D. in Computer Science and Engineering from University of North Texas. Her research interests include modeling and control of large-scale dynamical systems and spatiotemporal data management. Her email address is junfei.xie@tamucc.edu.

CHENYUAN HE is now working toward the Ph.D. degree at UTA. Her research interest lies in big spatiotemporal data-driven decision-making. Her email address is chenyuan.he@mavs.uta.edu.

YAN WAN is an Associate Professor from University of Texas-Arlington (UTA). She holds a Ph.D. in Electrical Engineering from Washington State University. Her research interests lie in the modeling, evaluation, and control of large-scale networks. Her email address is yan.wan@uta.edu.

KEVIN MILLS is a senior research scientist at NIST. He received the Ph.D. degree from George Mason University in 1996. His research interests include studying macroscopic behavior in complex information systems, such as the Internet and computational clouds. His email address is kmills@nist.gov.

CHRISTOPHER DABROWSKI is a computer scientist at NIST. He received the M.S. degree from Johns Hopkins University in 1983. His research interests include complex computer networks as well as early warning signals associated with network collapse. His email address is cdabrowski@nist.gov.