Cluster-Based Spatiotemporal Background Traffic Generation for Network Simulation

TING LI and JASON LIU, Florida International University

To reduce the computational complexity of large-scale network simulation, one needs to distinguish foreground traffic generated by the target applications one intends to study from background traffic that represents the bulk of the network traffic generated by other applications. Background traffic competes with foreground traffic for network resources and consequently plays an important role in determining the behavior of network applications. Existing background traffic models either operate only at coarse time granularity or focus only on individual links. There is little insight on how to meaningfully apply realistic background traffic over the entire network. In this article, we propose a method for generating background traffic with spatial and temporal characteristics observed from real traffic traces. We apply data clustering techniques to describe the behavior of end hosts as a function of multidimensional attributes and group them into distinct classes, and then map the classes to simulated routers so that we can generate traffic in accordance with the cluster-level statistics. The proposed traffic generator makes no assumption on the target network topology. It is also capable of scaling the generated traffic so that the traffic intensity can be varied accordingly in order to test applications under different and yet realistic network conditions. Experiments show that our method is able to generate traffic that maintains the same spatial and temporal characteristics as in the observed traffic traces.

Categories and Subject Descriptors: I.6.3 [**Simulation and Modeling**]: Applications; I.6.5 [**Simulation and Modeling**]: Model Development

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Network simulation, background traffic model, spatiotemporal network traffic characteristics, network traffic clustering

ACM Reference Format:
Ting Li and Jason Liu. 2014. Cluster-based spatiotemporal background traffic generation for network simulation. ACM Trans. Model. Comput. Simul. 25, 1, Article 4 (November 2014), 25 pages.
DOI: http://dx.doi.org/10.1145/2667222

1. INTRODUCTION

The ability to generate representative traffic is crucial for network simulators, emulators, and other empirical testbeds to effectively evaluate next-generation network protocols and applications. It is a nontrivial task to model Internet traffic, given the scale and diversity of today's applications and the sophistication of user behaviors.

In order to reduce the computational complexity of network simulation, we need to make a distinction between the *foreground traffic*, which is generated by the target

The research is supported in part by NSF grants (CNS-0836408, CCF-0937964, HRD-0833093) and two subcontracts from the GENI Project Office at Raytheon BBN Technologies (NSF CNS-0714770 and CNS-1346688).
Authors' address: T. Li and J. Liu, School of Computing and Information Sciences, Florida International University, 11200 SW 8th St, Miami, FL 33199; emails: {tli001, liux}@cis.fiu.edu.

DOI: http://dx.doi.org/10.1145/2667222

applications the researchers intend to study and therefore must be simulated with high fidelity, and the *background traffic*, which represents the bulk of the network traffic generated by other applications that do not require detailed models. Although the applications that produce the background traffic are of secondary interest, their behavior has a significant impact on the target applications, since the background traffic competes with the foreground traffic for network resources and therefore can drastically affect the behavior of the target applications.

Due to the diversity and complexity of today's network traffic, there is little agreement in the community on the proper use of background traffic in network experiments and performance evaluation studies. We can divide the existing traffic models into spatial and temporal models. *Spatial models* distribute traffic onto the network links based on traffic matrices. They generally focus on aggregate traffic intensity (rather than individual flows or packets) and can only deal with traffic variations at coarse timescales (e.g., in minutes). As a result, they may not be able to accurately capture the interaction with the foreground traffic, represented normally as individual packets or flows. *Temporal models* are based on packet traces collected for individual links; they can generate traffic as individual flows or packets, and therefore are able to capture the interaction with the target applications more accurately. Temporal models, however, can only work with individual links or paths; they are not able to represent the spatial distribution of traffic over the entire network. To the best of our knowledge, there is no existing background traffic model that can produce network-wide traffic on arbitrary network topology and simultaneously maintain the spatial and temporal characteristics of the observed Internet traffic.

In this article, we aim to provide a spatiotemporal background traffic generator that can be easily applied for network studies. We expect that the traffic generator should meet the following criteria:

—*Spatiotemporal correlation*: The traffic generator should jointly consider both spatial and temporal structures; that is, the generator needs not only to reproduce the bursty traffic behavior as observed on individual links and/or paths but also to reasonably place traffic on the target network topology so as to capture the spatial distribution of traffic covering the entire network.
—*Realism*: The traffic generator should be based on real traffic traces and traffic matrices, whenever available, in order to accurately represent the global Internet traffic behavior that constantly changes over time.
—*Flexibility*: The traffic generator should be flexible: the generator needs to produce traffic with various traffic conditions, for different test scenarios, and on arbitrary network topologies, in order to evaluate the target applications. That is, the traffic generator needs to provide the necessary "control knobs"—the ability to tune certain model parameters—while keeping important spatial and temporal traffic characteristics invariant in order to maintain a good level of realism.

To derive the spatiotemporal background traffic model, we start by analyzing the traffic behavior observed from the real network (i.e., using traffic traces). We adopt clustering techniques to classify the traffic in order to efficiently and effectively discover the underlying traffic patterns. More specifically, we describe the traffic as a function of multidimensional attributes and then apply a clustering algorithm to group the end hosts according to their contribution in the observed traffic. Different from other traffic clustering approaches, we carefully choose the features that define the clusters to facilitate traffic generation. The resulting traffic classes are then mapped onto a given network topology. This is achieved by first stochastically determining an origin–destination (OD) traffic matrix for the given network and then by overlaying the traffic sources and destinations belonging to the traffic classes onto the network according

to the traffic matrix. Once mapped, the traffic sources and destinations are able to populate the network with traffic according to a stochastic arrival process.

The novelty of our approach can be summarized as follows. Our method uses the clustering technique for background traffic modeling and simulation. By classifying traffic using the multidimensional attributes, we are able to effectively discover and succinctly summarize the network traffic patterns using cluster-level characteristics. By judiciously distributing the cluster-level traffic sources and destinations, we are able to spread the traffic across the entire network while maintaining the underlying spatial structure. By conducting the traffic flows in accordance with cluster-level statistics, we are able to maintain the temporal structure of the traffic flowing through the network links. By providing the mechanism for scaling the traffic intensity level on the network links while maintaining the same spatial and temporal characteristics, we are able to test applications under diverse and yet realistic traffic conditions.

We conducted experiments to validate our cluster-based method for spatiotemporal background traffic generation. The results show that the generated traffic is able to maintain the spatial structure in terms of link utilization and traffic distribution. It also maintains the temporal structure; the generated traffic is statistically similar to the original traffic traces. The proposed method is not limited to network simulation; we expect that our spatiotemporal background traffic generation method can be applied for network emulation as well as in empirical studies. For example, using the symbiotic simulation method [Erazo and Liu 2013], one can use simulation to generate spatiotemporal background traffic to influence the foreground traffic generated by real applications running in an emulated environment. In this article, however, we focus only on the simulation aspects.

The rest of the article is organized as follows. We describe the related work in Section 2. In Section 3, we provide an overview of our cluster-based method for spatiotemporal background traffic generation. In the subsequent sections, we describe the details of the proposed method. We conducted experiments to validate our model; the results are presented in Section 7. In Section 8, we outline our conclusion and discuss future work.

2. BACKGROUND

In this section, we first review existing traffic models, including temporal, spatial, and spatiotemporal models. We then provide a brief summary of related work in traffic classification. Finally, we describe the current state of using background traffic in network studies by conducting a survey for papers in SIGCOMM 2007–2013.

2.1. Existing Traffic Models

There are many existing efforts for network traffic generation. We divide them into temporal models, spatial models, and spatiotemporal models.

Temporal models include detailed packet traffic generators, such as Harpoon [Sommers and Barford 2004], NTools [Vegh 2013], Surge [Barford and Crovella 1998], Swing [Vishwanath and Vahdat 2006], and Tmix [Weigle et al. 2006]. These models analyze existing packet traces and subsequently generate traffic at the packet level. An important limitation of temporal models is that they focus only on an individual link (such as the bottleneck link in a dumbbell topology) or a specific path between two end hosts. The traffic cannot be extended easily to cover the entire network in order to be used as background traffic for studying applications distributed over an arbitrary network topology. It should be noted that generating traffic packet by packet is accurate but can also be computationally expensive for background traffic generation in simulation. Background traffic represents the bulk of the network traffic and yet it is not the focus of a simulation study. To reduce the computational complexity, abstract

models can be used to describe traffic at the flow level using fluids (e.g., [Liu et al. 2001; Misra et al. 2000, 2003; Kesidis et al. 1996; Nicol 2001; Nicol and Yan 2004; Ahn and Danzig 1996; Guo et al. 2000; Li et al. 2013]).

Spatial models distribute traffic based on traffic matrices. A traffic matrix represents the aggregate volume of traffic flowing between all OD pairs within a certain time interval. A traffic matrix is usually used as input for network design or for network management, such as capacity planning and traffic engineering. There has been extensive work on estimating the traffic matrix. The gravity model assumes that the traffic between an origin–destination pair is proportional to the total traffic from the source node to the destination node [Zhang et al. 2003a]. The main drawback is that the model assumes independence between the source and destination. To solve this problem, generalized gravity models extend the gravity model by separating traffic into classes and applying the gravity model on each class of traffic [Zhang et al. 2003b, 2005b]. The discrete choices model (DCM) is also a variation of the gravity model, which is based on the choice models for decision behavior, where the ingress nodes make decisions on the traffic volume and traffic destination based on user behavior and network configuration [Medina et al. 2002]. The independent connections model (ICM) focuses on connections between the traffic initiators and traffic responders; it considers the forward traffic proportion, the activity level of the users at a node, and the preference of a node as the peer of a connection, assuming independence between the connections [Erramilli et al. 2006]. Both DCM and ICM require a large number of parameters to achieve accuracy. The low-rank model provides a simpler and yet more general model, which can be treated as a weighted sum of gravity models [Bharti et al. 2010]. All of these spatial models focus only on the spatial distribution of traffic; the resulting traffic intensity remains constant during a given time interval, the size of which is usually predetermined according to the measurement requirements and is normally in minutes or larger. At this time granularity, one cannot accurately study the effect of background traffic on the individual packets generated by the foreground applications.

Spatiotemporal models consider both temporal and spatial structures. There are methods for traffic matrix estimation that also focus on time-dependent properties. Although we treat them here as spatiotemporal models, they are not really concerned with the spatiotemporal correlations of the traffic. Roughan et al. [2002] proposed a temporal model for the OD flows traversing backbone routers. The traffic intensity is modeled with four components: a long-term trend that captures the overall traffic behavior over a long period of time, a seasonal (cyclical) component that describes any periodic behavior in the traffic, a random fluctuation component that models the small fluctuation of the traffic, and an anomaly component that models the large variation of traffic from anomalies. Fourier analysis can be used to capture the periodic nature of the OD flows by representing the cyclical signal with a small number of Fourier coefficients [Tune and Roughan 2003]. Wavelets are also used to capture both short-range and long-range dependencies [Papagiannaki et al. 2003; Abry and Veitch 1998]. Principal components analysis (PCA) can be used to describe the OD flows by using a small number of "eigenflows" [Lakhina et al. 2004]. All these methods are data-driven methods and rely heavily on measurements. However, they deal with aggregate flows, which may not be able to achieve the level of accuracy necessary to capture the interactions with the foreground traffic. The derived temporal characteristics of the traffic are also independent of the spatial distribution.

There are two recent papers on the spatiotemporal correlation. Zhang et al. [2009] proposed a method that represents the traffic matrix by the low-rank approximation and uses a rank model to approximate both spatial and temporal correlations of the traffic matrix. The drawback of this model is that it is difficult to interpret the model

parameters; they are not directly related to network aspects or user behaviors. This also makes it difficult to tune the model for simulation purposes. Sommers et al. [2011] proposed an interesting method for network-wide flow record generation. The method is designed to generate representative benign and anomalous traffic, especially useful for anomaly detection. It builds on the Harpoon traffic generator [Sommers and Barford 2004] to allocate sources and destinations for traffic flows. Harpoon assigns weights to the IP addresses from a selection pool; the weights can be determined by the empirical distribution observed from the real network. The method proportionally selects the IP addresses according to the number of connections they involve; however, it does not consider the spatial distribution of the IP addresses and the connectivity between the IP addresses. In this case, for example, the method may not be able to reflect the existence of hotspots in the network. Comparatively, our method takes user behavior, node distribution, network connectivity, and flow-level statistics into consideration. As a result, we expect that our method is able to capture the spatial and temporal correlations of the traffic over the entire network.

2.2. Traffic Classification

Our method uses clustering techniques to classify traffic. There has been extensive work on traffic classification using machine-learning techniques. We roughly group the traffic classification methods into three types. The first type of methods (e.g., [Roughan et al. 2004; McGregor et al. 2004]) classify traffic based solely on flow-level statistics, such as traffic volume and packet size, without considering the end-user behavior. The second type of methods (e.g., [Karagiannis et al. 2005; Xu et al. 2005]) classify traffic based on end-user behavior but remain indifferent to network dynamics (such as network congestion and delays). The third type of methods (e.g., [Wei et al. 2006]) consider both end-user behaviors and network dynamics for classification. Our traffic model incorporates traffic classification belonging to this category.

Most existing traffic classification methods are used for analyzing traffic and detecting traffic anomalies, not for traffic generation. Valgenti and Kim [2012] proposed a traffic content generative mode that uses clustering techniques to determine the role of end hosts as either content providers or content consumers, and in doing so can generate traffic representative of content distribution over the network. Content generation is important for applications such as intrusion detection; however, their method does not consider traffic intensity, which is an important aspect for background traffic. In this study, we focus on workload-based background traffic generation. Content-based traffic generation needs to be application specific; we defer that to future work.

2.3. Use of Background Traffic in Network Experiments

To better understand the current use of background traffic in network studies, we conducted a survey of the SIGCOMM papers in the last 7 years (2007–2013). We categorize the papers that involve experiments with infrastructural networks according to their evaluation methods, which include real testbeds (R), simulation (S), emulation (E), or a combination of them. Table I shows the results. The "other" category consists of work that does not involve experiments with infrastructure networks, such as wireless communications, or includes only theoretical analyses.

We observe that the use of real testbeds and simulation accounts for a large proportion of the evaluative work. They are often used together with complementary roles. In a common scenario, the researchers use simulation to evaluate key functions under various network conditions and then use real testbeds, such as PlanetLab or other controlled platforms, including lab machines, university networks, and enterprise networks, to test real-world operations. In another common scenario, the researchers use a

Table I. SIGCOMM Papers in Different Categories

Year	R	S	E	R+S	S+E	S+R+E	Other	Total/yr
2013	12	8	1	6	1	0	9	37
2012	5	5	1	7	0	0	13	31
2011	8	4	0	5	0	0	12	29
2010	8	2	1	9	1	0	9	30
2009	12	2	1	5	0	0	8	28
2008	14	5	0	7	1	0	9	36
2007	13	5	1	7	0	1	8	35
Total	72	31	5	46	3	1	68	226

Table II. Use of Synthetic Background Traffic in Some SIGCOMM Papers

Paper	Network Topologies	Traffic Types
Probabilistic early response TCP [Bhandarkar et al. 2007]	Simple topologies with single and multiple bottlenecks	Long-lived TCP
Service differentiation [Podlesny and Gorinsky 2008]	Simple topologies with single and multiple bottlenecks	Long-lived TCP sampled flows
Scalable Ethernet architecture [Kim et al. 2008]	Campus network	Trace playback
Network measurement [Papageorge et al. 2009]	Simple topology with single bottleneck (dumbbell)	CBR traffic generator
Network-wide redundancy elimination [Anand et al. 2009]	Rocketfuel topologies	Gravity model
Denial of service [Liu et al. 2010]	Simple topologies with single and multiple bottlenecks	Long-lived TCP sampled flows
Flow-level measurement [Lee et al. 2010]	Simple topology with single bottleneck (dumbbell)	Sampled flows traffic generator
Route reconfiguration [Wang et al. 2010]	RocketFuel topologies US-ISP, GT-ITM, Abilene	Gravity model trace playback
Protocol manipulation attacks [Kothari et al. 2011]	Simple topology with single bottleneck (dumbbell)	Long-lived TCP
Flexible transport protocol [Han et al. 2013]	Simple topologies with single and multiple bottlenecks	Sampled flows

real testbed to conduct small-scale studies and then resort to simulation for large-scale experiments.

In both cases, simulation offers unique capabilities for network experimentation. Next, we specifically focus on simulation studies that require network traffic for evaluation. Table II lists the papers that use synthetic network traffic in experimental studies. We observe that various background traffic models have been used, including long-lived TCP flows, constant-bit-rate (CBR) traffic, packet trace playback, sampled flows, and traffic generators. Apparently, long-lived TCP and CBR are limited in terms of representing the temporal behavior of the Internet traffic, such as traffic burstiness caused by long-term dependencies. In order to better capture the temporal structure of the traffic demand, people resort to either using direct playback of the packet traces or applying empirical sampling of the packet traces to obtain the random flow interarrival times and flow lengths (we name this method "sampled flows" in the table).

Only a few studies involve existing traffic generators, and most of them are limited to simple network topologies, such as dumbbell. For studies that require more sophisticated network topologies, there are no commonly adopted methods for obtaining the background traffic. Anand et al. [2009] applied the gravity model to estimate the traffic matrix at the PoP level (they use the RocketFuel ISP PoP-level topologies [Spring et al. 2004]), assuming the traffic is uniformly distributed among the access routers within each PoP. Wang et al. [2010] actually used the CAIDA trace to reproduce the

packet-level traffic flows. However, the derived temporal behavior of their traffic would be independent of the spatial distribution.

From this survey, we see that there is a significant lack of network-scale background traffic models in network experimentation.

3. OVERVIEW OF CLUSTER-BASED SPATIOTEMPORAL TRAFFIC GENERATION

This section presents an overview of our cluster-based method for spatiotemporal traffic generation. The method makes two assumptions. The method is based on network measurements; in particular, it applies statistical analysis to a packet trace collected at a specific network link. Here, we assume that the user behavior observed from the packet trace is representative and can reveal the network-wide traffic pattern. Our results could be improved if using network traces from several vantage points (e.g., for different link types) to provide a broader view of the overall network traffic. We will explore the use of multiple traces for traffic generation in future work. We also assume that traffic engineering can perfectly balance the traffic load among all links of a given network topology. This would allow us to extend the measurements from a specific link to the entire network. The assumption is generally true; however, it does not consider cases where traffic may be skewed temporarily on some links. One can implement methods to intentionally and probabilistically create traffic load imbalance. We will explore this issue in the future.

Our method is divided into three steps:

(1) We analyze the traffic trace by characterizing the traffic as a function of multidimensional attributes and then grouping the end hosts with similar features into clusters. In this way, we can identify the unique characteristics of different groups of end hosts, such as traffic hotspots (either sources or sinks that generate large data volumes) and heavily connected servers. This high-level traffic behavior is then summarized as flow-level statistics between the clusters. The result will be used subsequently for traffic generation.
(2) Given a network topology that consists of routers, we need to associate them with the clusters derived from the previous step. Because we are only interested in the interaction between the foreground and background traffic at the network links connecting the routers, the traffic generator only needs to produce traffic on those links, as opposed to modeling each individual end host. A router presumably can connect to many end hosts, each belonging to a different cluster. That is, a router may simultaneously belong to multiple clusters. The goal of this step is to "reasonably" distribute (or map) the clusters to the routers of an arbitrary network topology.
(3) Given the network topology, the cluster-level traffic summary, and the mapping from clusters to routers, we can now generate traffic by randomly creating flows between selected sources and destinations according to the statistical results.

In the following sections, we separately present the details for each of the three steps of our proposed method.

4. STEP 1: TRAFFIC CLASSIFICATION

To generate realistic traffic, we first analyze existing traffic traces collected from Internet measurement points, cluster the end hosts using multidimensional attributes, and then summarize the high-level traffic behavior between the clusters.

4.1. Traffic Traces

To describe our method, we select three traffic traces obtained from the public Internet data repositories as examples. The traces are collected at distinct Internet vantage

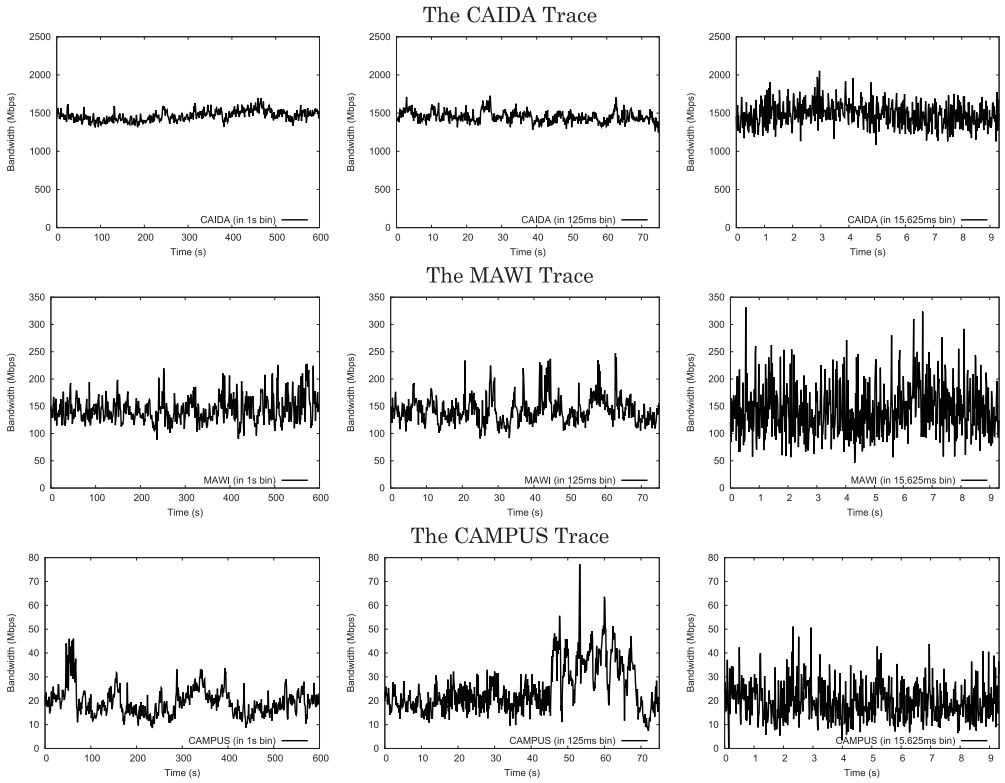

Fig. 1. Traffic intensity at different timescale for the three traces.

points. They include a CAIDA trace, collected from a U.S. backbone link [CAIDA 2011]; a MAWI trace, collected from a trans-Pacific link [MAWI 2013]; and a campus network trace, collected at the uplink from a university [Dainotti et al. 2008, 2009]. More specifically, the CAIDA trace was captured in July 2011 at the equinix-chicago Internet data collection monitor in Chicago from a 10GigE backbone link of a Tier-1 ISP connecting between Chicago and Seattle. The MAWI trace was collected in May 2013 at sample point F from a 150Mbps trans-Pacific link. The CAMPUS trace we use was the web traffic generated by clients inside the University of Napoli "Federico II" network in June 2004. The trace was collected at the campus's 200Mbps uplink connecting the network to the rest of the Internet.

For the traffic analysis shown later, we used only the first 10 minutes of each trace. Also, since we conduct flow-level analysis, we limit traffic to TCP only. This should be fine as we have observed that TCP is the dominant traffic in all traces: it is over 81% for the CAIDA trace, over 83% for the MAWI trace, and 100% for the CAMPUS trace (since the trace consists of only web traffic). The 10-minute CAIDA trace contains over 5.8M flows, 168M packets, and 804K distinct IP addresses. The MAWI trace contains approximately 510K flows, 13.5M packets, and 15K distinct IP addresses. The CAMPUS trace contains about 100K flows, 14M packets, and 4K distinct IP addresses.

Figure 1 shows the traffic intensity of the three traces, each in a separate row. At each row, from left to right, we decrease the sampling interval while maintaining the number of samples at 600. The starting sampling interval is 1 second; each subsequent plot is obtained by randomly choosing a subinterval, the length of which is one-eighth

of the previous one. The x-axis of each plot represents the time offset from the starting time of the randomly chosen subintervals. The figure shows intuitively the scale-free traffic behavior at different time resolutions.

4.2. Clustering End Hosts

For traffic clustering, we focus on four features (or attributes) for each end host, represented by a distinct IP address that appears in the traces: (1) the number of flows (or TCP connections) involving the end host, (2) the number of distinct peers connected with the end host, (3) the total number of bytes sent from the end host, and (4) the total number of bytes received by the end host. We obtain these attributes by analyzing the packet length, the source and destination IP addresses, and the TCP flags from the packet headers. We expect these attributes are enough to reveal the hidden spatial distribution underpinning the user access patterns, as well as the temporal behavior of individual traffic flows.

We conduct k-means clustering [MacQueen 1967], in particular, using a data mining software, called WEKA [Hall et al. 2009]. K-means is a simple unsupervised learning algorithm. It aims to partition the observations into k clusters: an observation is a member of a cluster if it has the closest distance to the centroid of the cluster. In our case, a trace often consists of many flows involving a large number of IP addresses. We classify these IP addresses into a small number of clusters expecting that the components within each cluster behave similarly. We note that the values of some of the attributes may differ significantly in magnitude. For example, the number of flows involving a particular IP address ranges from one to several hundreds; the total number of bytes sent or received by an IP varies from a few bytes to several megabytes or more. For these attributes, we take logarithmic values for clustering following a common practice.

A main concern of the k-means algorithm is that the number of clusters, k, must be provided a priori. Several methods exist for determining the proper number of clusters needed for a given dataset. Increasing k would result in smaller errors but would also increase the computation. We use a popular method to choose k: we run the k-means clustering algorithm with different values of k and select the one such that the clustering error is around the "elbow"—the error decreases insignificantly when the number of clusters increases from that point.

Table III presents the detailed clustering results of the three traces, from which we can make the following observations:

(1) The clusters vary greatly in size (in terms of the number of distinct IP addresses).
(2) Some clusters operate as data generators, and some as data sinks; in both cases, significant asymmetry exists between the number of bytes sent and received by an IP address.
(3) Traffic intensity, that is, the average number of data sent and received per IP address, varies significantly between the clusters, suggesting the existence of hot spots in the network.
(4) Since different traces observe different traffic at different vantage points, the clustering results are different.

4.3. Cluster-Level Traffic Summary

Once we have determined the clusters, we can collect the statistics of the traffic flows between the clusters. In the following, we summarize the results, which we later use for traffic generation:

—Let k be the number of clusters. Let C_i be the set of distinct IP addresses (we treat them as individual users) in cluster i, for all $i \in \{0, 1, \ldots, k-1\}$. We calculate the

Table III. The Clustering Result

The CAIDA Trace

Cluster	0	1	2	3	4	5	6	7	8
IPs	33840 (4%)	232003 (29%)	114516 (14%)	39832 (5%)	36923 (5%)	77461 (10%)	71595 (9%)	89322 (11%)	108739 (14%)
Flows	23	1	2	10	7	2	11	7	1
Peers	5	1	1	4	2	1	3	3	1
Sent	38082	368	3322	0	0	0	3220	1801	0
Rcvd	0	0	0	2262	157456	4692	9464	0	278

The MAWI Trace

Cluster	0	1	2	3	4	5	6	7	8
IPs	2108 (15%)	678 (5%)	475 (3%)	3438 (24%)	1914 (13%)	1710 (12%)	2480 (17%)	517 (4%)	1202 (8%)
Flows	3	2	60	1	81	11	3	2	1
Peers	1	1	16	1	1	2	1	1	1
Sent	991	62643	100811	611	26995	4469	0	978	337
Rcvd	0	1825	90654	625	41386	4677	1746	150859	4825

The CAMPUS Trace

Cluster	0	1	2	3	4	5	6	7
IPs	674 (17%)	430 (11%)	412 (10%)	540 (13%)	490 (12%)	244 (6%)	714 (18%)	547 (14%)
Flows	1	32	12	59	4	235	4	2
Peers	1	1	4	8	2	25	1	1
Sent	462	230199	24147	75826	2995	331274	152940	21766
Rcvd	438	28825	27076	312731	3286	1052733	7963	763

population density for cluster i as

$$\phi_i = \frac{|C_i|}{\sum_{0 \le j < k} |C_j|}. \tag{1}$$

—We use F_{ij} to denote the total number of flows observed in the trace between an IP address in C_i and an IP address in C_j, where $0 \le i, j < k$. Note that for simplicity, we do not distinguish the direction of the flows. We count each flow in both directions: each flow observed in the trace between i and j is counted as 1/2 flows in F_{ij} and 1/2 flows in F_{ji}. Therefore, we have $F_{ij} = F_{ji}$. As a special case, F_{ii} is the number of flows between two IPs of the same cluster i. We use F to denote the total number of flows observed in the trace, which can also be expressed using:

$$F = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} F_{ij}. \tag{2}$$

We calculate the *flow density* for cluster i as the proportion of flows that involve an IP address in cluster i among all flows (including those within the same cluster):

$$\psi_i = \frac{\sum_{0 \le j < k} F_{ij}}{F}. \tag{3}$$

We also calculate the *peering probability* from cluster i to cluster j as the number of flows between cluster i and cluster j, divided by the total number of flows involving

cluster i:

$$\omega_{ij} = \frac{F_{ij}}{\sum_{0 \le x < k} F_{ix}}. \tag{4}$$

—Let T be the duration of the trace. We calculate the aggregate flow arrival rate from cluster i to cluster j, in number of flows per second, as follows:

$$\lambda_{ij} = \frac{F_{ij}}{T}. \tag{5}$$

Note that since we do not distinguish direction of the flows, $\lambda_{ij} = \lambda_{ji}$. As a special case, λ_{ii} is the aggregate flow rate between IPs of the same cluster i. Here we assume that the flow arrival is a stationary Poisson process. To capture the nonstationary behavior (such as the diurnal effect), we can easily extend this method by using a piece-wise constant flow arrival rate for each time interval.

—We describe the flow size (in number of bytes) from cluster i to cluster j using a log-normal distribution with parameters μ_{ij} and σ_{ij} (we show evidence momentarily). Here we use the flow size distribution to capture the asymmetric behavior of traffic between clusters. For each identified flow in the trace, we separate the flow into two flows, one for each direction. If the packet's source address belongs to cluster i and its destination address belongs to cluster j, we add the packet size to the flow size from i to j, and vice versa. Note that the size of the directed flows is in general asymmetric. Once we know the mean, m, and the variance, v, of the size of the directed flows, it is easy to calculate the parameters of the log-normal distribution:

$$\mu = \ln\left(\frac{m^2}{\sqrt{v + m^2}}\right), \quad \sigma = \sqrt{\ln\left(1 + \frac{v}{m^2}\right)}. \tag{6}$$

—We calculate the aggregate traffic intensity from cluster i to cluster j, in number of bytes per second, as the product of the aggregate flow arrival rate and the mean flow size:

$$\delta_{ij} = \lambda_{ij} \cdot e^{\mu_{ij} + \frac{\sigma_{ij}^2}{2}}. \tag{7}$$

The total inflow traffic intensity and outflow traffic intensity at cluster i, also in number of bytes per second, can be summed up easily:

$$\delta_{\bullet i} = \sum_{j=0}^{k-1} \delta_{ji}, \quad \delta_{i\bullet} = \sum_{j=0}^{k-1} \delta_{ij}. \tag{8}$$

Note that both inflow and outflow traffic intensity also include traffic going between end hosts in the same cluster.

Log-normal distribution is appropriate for describing the flow size. Figure 2 shows the Q-Q plots of the flow size between two selected clusters from the CAIDA, MAWI, and CAMPUS traces independently against a log-normal distribution with parameters estimated from the trace data. They match well. We observe similar results between all clusters for the three traces, although they use different log-normal parameters.

5. STEP 2: MAPPING CLUSTERS TO ROUTERS

The generated traffic will be conducted between end hosts. Given a simulated network topology that consists of connected routers, it is possible that we attach the end hosts to the routers and then have the end hosts produce the traffic accordingly. However, this would be unnecessary since we are only interested in generating the background

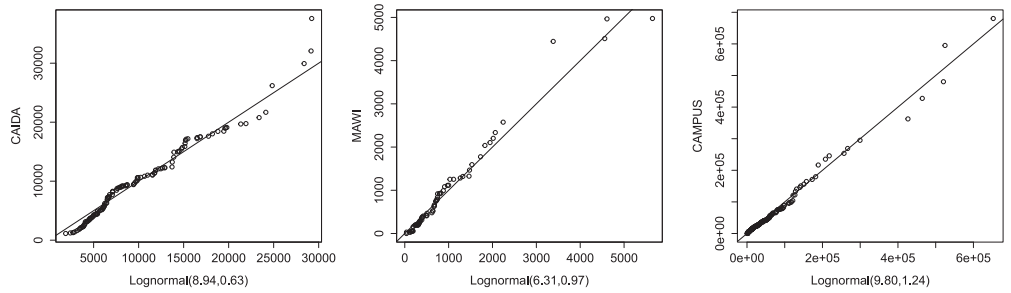

Fig. 2. Q-Q plot of flow size versus log-normal for the three traces.

traffic on the links between routers of the given topology. In this case, one can simply have the routers assume the role of traffic sources and destinations and produce the traffic among them accordingly. Since each router may direct traffic that belongs to the attached end hosts of different clusters, we may need to map the clusters to multiple routers of the given topology to preserve the spatial distribution of the traffic.

Doing so would require information on the spatial distribution of traffic among the routers. For real networks, this information would be mostly proprietary and therefore difficult to obtain. For synthetic network topologies used in simulation, it is impossible. One would have to make certain assumptions. For example, one could assume that traffic flows are uniformly distributed over the network. This is unrealistic, however, since it does not consider the network effect, such as link bandwidths, delays, congestions, and routing. In this section, we present an algorithm that can reasonably map the clusters to the routers, assuming that the traffic should be distributed over the network as evenly as possible, as long as it will not load any particular link disproportionately, and the traffic load on any link will not exceed its link capacity.

Our solution is divided into two steps. In the first step, we derive the traffic matrix of a given network. Here, we apply an existing traffic matrix estimation technique, which can stochastically determine the traffic matrix for arbitrary network topologies. In the second step, we assign the end hosts belonging to different clusters to the routers in the given topology so that the resulting traffic is compatible with the traffic matrix obtained from the first step.

5.1. Deriving Traffic Matrix for Arbitrary Network Topology

We first derive the traffic matrix for any arbitrary topology, which consists of n routers and m links.[1] The goal is to calculate the traffic intensity $r_{i,j}$ from router i to router j, for all $0 \leq i, j < n$, and $i \neq j$. We adopt the method proposed by Nucci et al. [2005] to estimate the traffic matrix. Their method first samples the flow rates from a statistical distribution observed from measurement. The sampled rates are then assigned to the source–destination pairs of the network by solving an optimization problem. In the following, we describe only the formulation of the problem specific to our approach. We refer the readers to the original paper for further details [Nucci et al. 2005].

Sampling Random Flow Rates. We assume that the flow rates follow the log-normal distribution. It has been shown that log-normal distribution provides the best fit for both Sprint and Abilene networks [Nucci et al. 2005]. For a given network topology and routing information (we assume static routing), we first determine the log-normal parameters μ and σ.

[1]We only need to consider access routers that we assume are the ingress and egress points of all background traffic.

Suppose μ_c and σ_c are the mean and standard deviation of the link capacity of the given network. From static routing information, we can find the path length, π_{ij}, in number of links from router i to router j. We can then calculate the average number of source–destination flows that traverse each link in the network:

$$\gamma = \sum_{\substack{0 \le i,\, j < n \\ i \ne j}} \frac{\pi_{ij}}{m}.$$

We assume that on average, we should maintain the same link utilization as observed in the packet trace. Let ρ be the link utilization of the observed packet trace, which can be calculated as the total amount of data transferred over the link divided by the duration of the trace and the link capacity. We can calculate the mean of the log-normal distribution for the flow rate to be $\mu_c \rho \gamma^{-1}$. That is, we scale the mean link capacity by a factor of $\rho\gamma^{-1}$. If we scale the standard deviation by the same factor, we can expect it to be $\sigma_c \rho \gamma^{-1}$. Therefore, we can calculate the parameters for the log-normal distribution, following Equation (6):

$$\mu = \ln\left(\mu_c^2 \rho/\gamma\right) - \frac{1}{2}\ln\left(\mu_c^2 + \sigma_c^2\right), \quad \sigma = \sqrt{\ln\left(1 + \sigma_c^2/\mu_c^2\right)}.$$

Given μ and σ, we take a sample of size $n(n-1)$ from the log-normal distribution; we denote the sampled flow rates as $\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_{n(n-1)-1}$.

Integer Linear Programming (ILP). Next, we simply follow the method proposed by Nucci et al. [2005], which formulates the problem as an optimization problem that can be solved using Integer Linear Programming (ILP). The result is that sampled rates are assigned to the $n(n-1)$ source–destination pairs so that it minimizes the maximum link utilization. This is a reasonable goal since network traffic engineering is widely used by ISPs to balance traffic load and minimize congestion.

The output of the solution is a set of mapping indicators $\mathcal{I}_p^{ij}$, where $0 \le p < n(n-1)$, $0 \le i, j < n$, and $i \ne j$. The indicator is 1 if flow rate $\mathcal{A}_p$ is assigned to the source–destination pair from router i to router j, and 0 otherwise. Finally, we can obtain the traffic matrix, where the traffic intensity from router i to router j can be obtained as follows:

$$r_{ij} = \begin{cases} \sum_{p=0}^{n(n-1)-1} \mathcal{I}_p^{ij} \mathcal{A}_p & \text{if } i \ne j \\ 0 & \text{otherwise.} \end{cases}$$

5.2. Solving Cluster-to-Router Mapping

In step 1, we obtain the cluster-level traffic summary from the traffic trace. In the previous section, we obtained a traffic matrix, which represents the traffic demand between the routers of any given network topology. In this section, we present an algorithm to associate the clusters to the routers. More specifically, we need to solve for p_{si}, which is the proportion of the end hosts belonging to cluster s that are mapped to router i, where $0 \le s < k$ and $0 \le i < n$.

For a given network topology, we define the *user density* of router i as d_i, where $0 \le d_i \le 1$ and $\sum_{0 \le i < n} d_i = 1$. A router's user density is a user-defined value; it is expected to be proportional to the number of end-users attached to the router. In cases where end-users' geographical distribution must be considered in the performance evaluation, this mechanism provides a way to distribute the traffic load accordingly. By default, one can simply assume a uniform distribution, that is, $d_i = n^{-1}$.

We note that there may be discrepancies between the amount of traffic over the network as specified by the traffic matrix and the amount of traffic shown in the

cluster-level traffic summary from the packet trace. This is normal. For example, the bandwidth could be significantly different between the links in the target network and the one from which we obtain the packet trace. To compensate for the difference, we define a traffic proportion factor, θ, as follows:

$$\theta = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} r_{ij}}{\sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \delta_{st}}. \tag{9}$$

The numerator is the total traffic flow rate reported by the traffic matrix; the denominator is the total traffic reported by the cluster-level traffic summary.

We formulate the problem as a quadratic programming problem. The goal is to find p_{si}, the proportion of end hosts in cluster s mapped to router i, so that we can maximize the spread of the traffic behavior that is represented by the clusters, that is, make the clusters distributed as evenly as possible, among all routers, subject to the constraints dictated by the given traffic matrix. The optimization problem can be formally specified as follows:

Minimize

$$\sum_{s=0}^{k-1}\sum_{i=0}^{n-1}\left(p_{si} - \frac{1}{n}\right)^2$$

subject to

$$\sum_{i=0}^{n-1} p_{si} = 1, \quad \forall s \in \{0, 1, \ldots, k-1\} \tag{10}$$

$$p_{si} \geq 0, \quad \forall s \in \{0, 1, \ldots, k-1\}, i \in \{0, 1, \ldots, n-1\} \tag{11}$$

$$\sum_{s=0}^{k-1} p_{si} \cdot \phi_s = d_i, \quad \forall i \in \{0, 1, \ldots, n-1\} \tag{12}$$

$$\theta \sum_{s=0}^{k-1} p_{si}\delta_{s\bullet} - \sum_{j=0}^{n-1} r_{ij} = \theta \sum_{s=0}^{k-1} p_{si}\delta_{\bullet s} - \sum_{j=0}^{n-1} r_{ji}, \quad \forall i \in \{0, 1, \ldots, n-1\} \tag{13}$$

$$\theta \sum_{s=0}^{k-1} p_{si}\delta_{s\bullet} \geq \sum_{j=0}^{n-1} r_{ij}, \quad \forall i \in \{0, 1, \ldots, n-1\}. \tag{14}$$

Equation (10) states that the proportion of end hosts of the clusters mapped to all routers should sum up to 1. Since they are proportions, Equation (11) states that they should not be negative. Equation (12) defines the user density at router i, d_i, which should be the sum of the proportion of end hosts of each cluster mapped to router i, p_{si}, multiplied by the cluster's population density, ϕ_s.

Equation (13) matches the traffic intensity observed by the cluster-level traffic summary with that specified by the traffic matrix at each router. The first term on the left-hand side of the equation is the sum of the outflow traffic intensity of all clusters assigned to router i, multiplied by the traffic proportion factor θ. The second term on the left-hand side of the equation is the total traffic sent from router i, as seen by the traffic matrix. The difference accounts for the traffic between the end hosts attached

to the same router i and therefore cannot be observed by the traffic matrix. Similarly, the right-hand side of the equation computes the difference between the sum of in-flow traffic intensity of all clusters assigned to the router and the total traffic received by the router as seen by the traffic matrix, which is also the traffic between the end hosts attached to the same router. Equation (14) makes sure that the difference is nonnegative.

The optimization problem is a convex quadratic programming problem with a positive definite objective matrix and therefore can be solved in polynomial time.

6. STEP 3: TRAFFIC GENERATION

From the first two steps, we have obtained the cluster-level traffic summary and a mapping from the clusters to routers for any given network topology. Now we are ready to generate the background traffic. Our method can be summarized as follows:

(1) We model the flow arrivals as a Poisson process (using exponentially distributed interarrival time), with an arrival rate

$$\lambda = \frac{\alpha F}{T}, \tag{15}$$

where F is the total number of flows observed in the packet trace (Equation (2)) and T is the duration of the trace. α is a scaling factor, a user-defined "control knob" for varying the intensity level of the generated background traffic in order to test applications under different network conditions. When $\alpha = 1$, we expect the traffic generator to generate network-wide traffic with similar traffic intensity as seen by the trace. If $\alpha = 2$ or $\alpha = 0.5$, for example, we expect the intensity of the generated traffic to be doubled or halved accordingly.

(2) For each flow arrival, we select the source cluster s with a probability equal to the cluster's flow density, ψ_s (Equation (3)).
(3) Select the source router i with probability p_{si}, which is the proportion of end hosts in cluster s mapped to router i.
(4) Select the destination cluster t using the peering probability, ω_{st} (Equation (4)).
(5) Select the destination router j with probability p_{tj}, which is the proportion of end hosts in cluster t mapped to router j.
(6) Create a TCP flow from router i to router j and transfer data of a certain amount; we sample the flow size (number of bytes) from the log-normal distribution with parameters μ_{st} and σ_{st}.
(7) The algorithm continues from step (2) for each new flow arrival.

In general, a background traffic generator does not need to take into consideration the source and destination IP addresses of the generated traffic flows. In some studies, however, one may need to preserve such information, for example, to allow simple packet inspection by an intrusion detection system.[2] To generate traffic between specific end hosts, one needs to first associate the IP addresses to the routers of the given topology. Suppose that N is the total number of distinct IP addresses we want to consider for background traffic generation. Each router i will have $(N \cdot d_i)$ IP addresses, where d_i is the user density of router i. We can assign these IP address to clusters based on the cluster distribution at this router. In particular, one can assign the IP

[2]As we mentioned earlier, content-based traffic generation is not the aim of this study. However, with this additional consideration, one can easily generate traffic flows between IP addresses that are distributed over the entire network.

Fig. 3. The backbone of the Abilene network.

addresses associated with router i to cluster c using the following proportion:

$$\kappa_{ci} = \frac{p_{ci} \cdot \phi_c}{\sum_{s=0}^{k-1} p_{si} \cdot \phi_s}, \tag{16}$$

where p_{si} is the proportion of the end hosts belonging to cluster s that are mapped to router i, and ϕ_s is the population density for cluster s. After we associate the IP addresses to the routers, we can generate the flows with specific source and destination IP addresses. More specifically, in step (3) of the previous algorithm, we can select the source address from all hosts attached to router i that belong to cluster s uniformly at random. Similarly, in step (5), we can select the destination address from all hosts attached to router j that belong to cluster t uniformly at random.

The major performance bottleneck of the traffic generation method is the same as the performance issue encountered by the traditional packet-oriented simulation. In a packet-oriented simulation, each packet-level instance such as packet arrival or packet departure is processed as a simulation event. The computational cost increases proportionally as the number of packets increases, which may result in an intolerable computational burden for large-scale network simulation. In contrast, fluid models model the traffic in terms of continuous fluid flows rather than individual packet instances. This model abstraction can significantly reduce the computational cost of the simulation by a factor of one to three orders of magnitude compared with the packet-oriented models (e.g., [Guo et al. 2000; Misra et al. 2000, 2003; Nicol 2001; Nicol and Yan 2004; Li et al. 2013]). We will investigate fluid-based background traffic generation in future work.

7. SIMULATION EXPERIMENTS

In this section, we validate our background traffic model, particularly focusing on the spatial and temporal properties of the generated traffic. We conduct the experiments under two simulation scenarios, one on a real backbone network and the other on a synthetic campus network.

7.1. The Abilene Network

We simulate the Abilene network to evaluate the basic properties of the generated traffic, especially its spatial distribution. The network, as shown in Figure 3, contains 12 routers and 15 links. The link connecting Indianapolis and Atlanta has a bandwidth

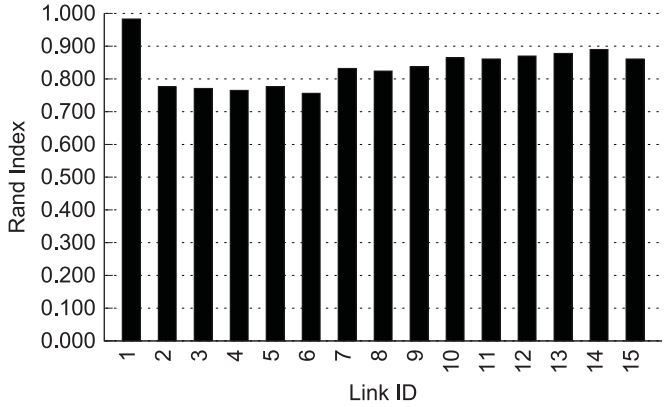

Fig. 4. Rand index of all links of the Abilene network.

of 2.5Gbps and all the other links have a bandwidth of 10Gbps. We use link ID 1 to 15 to identify the 15 links individually. The ID number of the 2.5Gbps link is 11. An important reason we decide to use this network is that the network has real traffic matrices available, which can help us determine whether our traffic model can properly preserve the spatial distribution. In the experiment, we use the same traffic matrix as the one described by Zhang et al. [2005a]. For packet trace, we use the CAIDA trace collected from a U.S. backbone link, as described in Section 4. In this experiment, since we only focus on the constitution and distribution of the background traffic, and since there is no foreground traffic to interfere with the generated background traffic, we simplify the traffic generator by replacing the computationally expensive packet-oriented simulation with a fluid model using constant-rate flows for expediency (at step 6 of the traffic generation algorithm).

First, we examine whether the generated background traffic would constitute the same type of flows as in the original packet trace. This can be achieved by applying the same clustering algorithm with the same set of attributes on the generated traffic trace and comparing clustering results with those from the original traffic trace. In particular, we use a metric, called "rand index," to determine the clustering similarity. The rand index measures the agreement between two clustering results by counting the membership of each point of the clustering. It has a value between 0 and 1, with 0 indicating that the two clusters do not agree on any pair of points and 1 indicating that the clusters are exactly the same. For each generated traffic trace (one for each link), let N be the total number of observations (i.e., distinct IP addresses) appearing in the trace (there are $N(N-1)/2$ observation pairs). If the pair of IP addresses is from the same cluster, we say the addresses are comembers. Let N_{11} be the number of observation pairs that are comembers in the original trace and still remain comembers in the generated traffic trace. Let N_{00} be the number of observation pairs that are not comembers in the original trace and still belong to different clusters in the generated traffic trace. We can define the rand index as follows:

$$S_R = \frac{2(N_{11} + N_{00})}{N(N-1)}.$$

Figure 4 shows the rand index for every link of the network. We see that the value never goes below 0.75, which indicates that the majority of the IP addresses maintain a similar membership association as in the original classification. That is, the constitution of the traffic matches well with the cluster assignment over the entire network when the traffic is spread among all links of the given network topology.

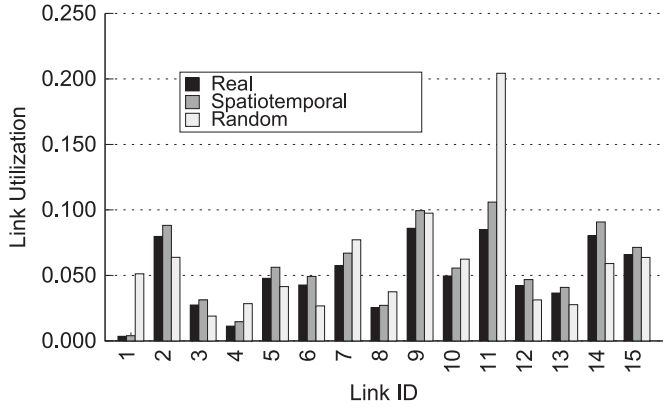

Fig. 5. Utilization of all links of the Abilene network.

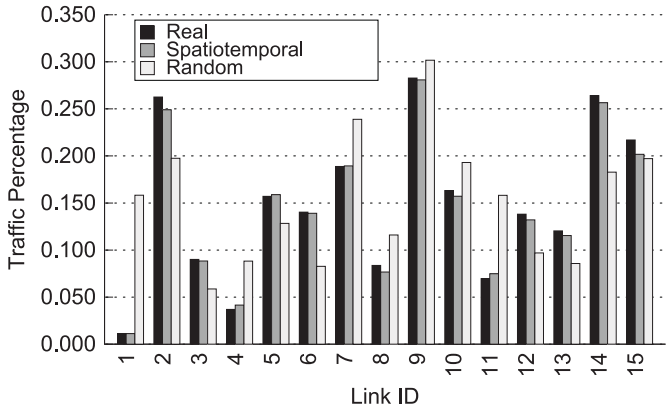

Fig. 6. Traffic distribution among all links of the Abilene network.

Next, we validate our traffic model by comparing the spatial distribution of the generated traffic against the real traffic matrix. In particular, we compare the link utilization and the traffic distribution (i.e., the percentage of traffic on the links) resulting from our spatiotemporal method (Spatiotemporal) against those from the original traffic matrix (Real). To make it more interesting, we also compare the results with those from using a method that places the flows over the network uniformly at random (Random). Figure 5 shows the link utilization on all 15 links, and Figure 6 shows the percentage of traffic on each of the 15 links. We can probably attribute the slight increase in the link utilization of our method to the TCP acknowledgments; in our method, we ignored the ACK flows. One could include the effect of the ACK flows by simply readjusting the traffic proportion factor in Equation (9). Overall, the results demonstrate that our spatiotemporal method is able to capture the spatial distribution of the overall traffic as dictated by the traffic matrix.

7.2. The Campus Network

In the following experiment, we use a synthetic campus network topology, as shown in Figure 7, which consists of 18 routers, among which 10 routers are access routers (marked with circles), which are connected with end hosts that generate the traffic. We designate different bandwidths to links so that we can observe the nonuniform

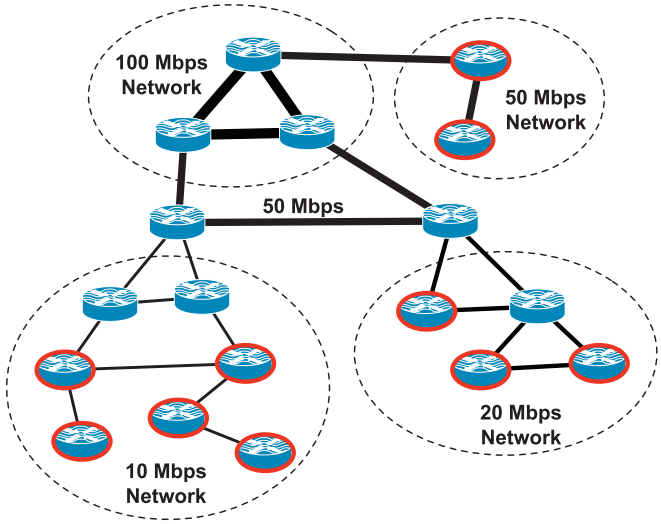

Fig. 7. A synthetic campus network.

traffic distribution over the network. For packet trace, we use the CAMPUS trace collected at a university campus unlink, as described in Section 4. Since we study the temporal characteristics of the generated background traffic, we use a detailed TCP implementation in simulation.

We examine the variation of intensity of the generated traffic over time on all links of the campus network. In the experiment, we set the scaling factor α to be 0.5, 1, 1.5, 2, 2.5, and 3, respectively, for different background traffic intensity levels. Figure 8 shows the traffic intensity on a 50Mbps link (we observed similar results for the other links as well). Each row shows the results using a different scaling factor. Similar to plots in Figure 1, at each row, from left to right, we decrease the sampling interval while maintaining the number of samples at 600. The starting sampling interval is 1 second, and each subsequent plot is obtained by randomly choosing a subinterval, the length of which is one-eighth of the one on the left. We see apparently that the traffic burstiness persists over different timescales regardless of the scaling factor.

To gain a more precise view of the traffic burstiness, we use the wavelet-scaling plot, also known as the energy plot, which can capture the correlation in the amount of traffic arriving at consecutive time intervals of a given size [Feldmann et al. 1999]. Figure 9 shows the energy plot of the generated traffic on the same 50Mbps link at different timescales and with different scaling factors. For comparison, we also plot the energy of the original CAMPUS trace. The x-axis shows a range of timescales, each being 2^{-j}, from $j = 0$ (1 second) to $j = 11$ (around 0.5ms). The y-axis shows the corresponding energy value in a logarithmic scale. A higher energy level represents more traffic burstiness. We see that the scaling factor has almost no impact on the traffic burstiness at different timescales. Furthermore, the generated traffic exhibits very similar burstiness when compared to the original packet trace, except when j is around 9 (at about 2ms), which is actually at a timescale not much larger than the packet transmission time. A slight deviation in the traffic burstiness at this small timescale would have little impact on the foreground traffic. For that reason, we chose to use fixed packet size in our traffic generator when generating the TCP flows.

While the scaling factor has little effect on the traffic burstiness, it is supposed to have significant impact on the traffic intensity, as is apparent in Figure 8. In

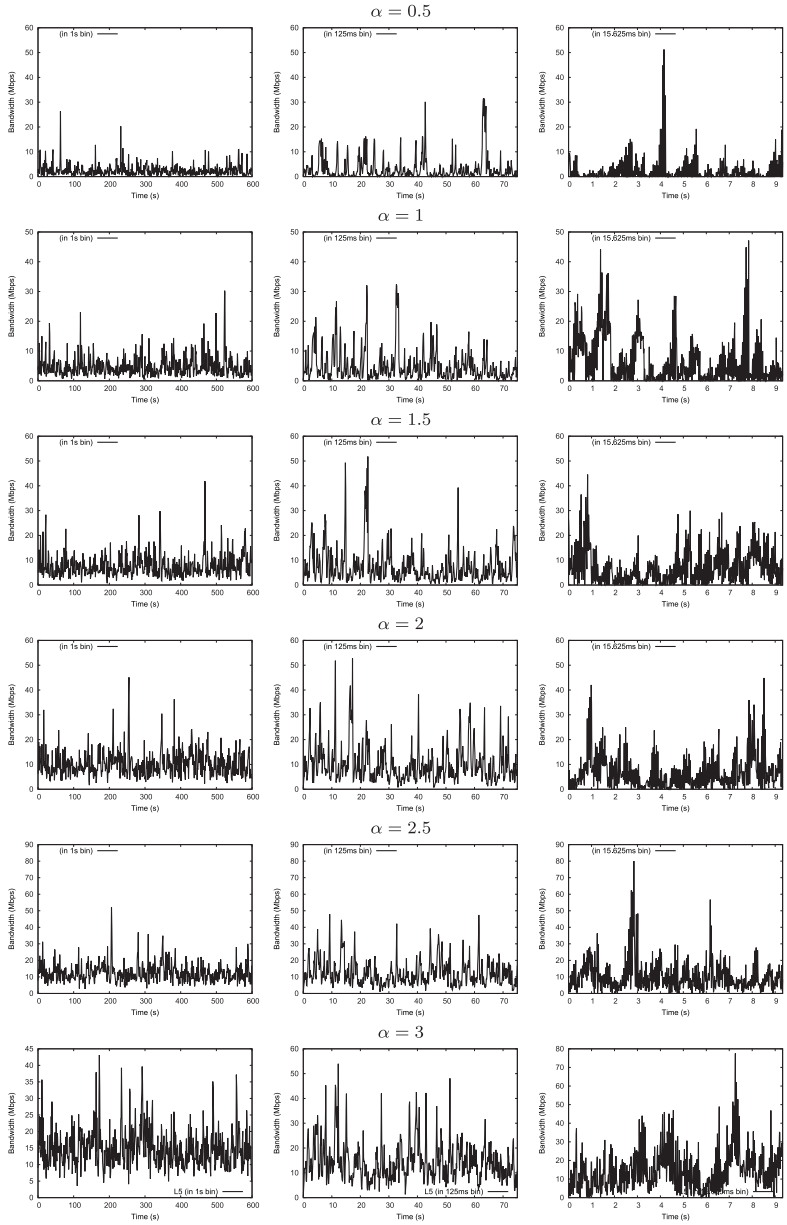

Fig. 8. Traffic intensity on one link of the campus network.

Figure 10, we show the link utilization with different scaling factors for all links that have background traffic.[3] We see that the traffic intensity increases almost proportionally on all links. Figure 11 shows the traffic distribution (the percentage of traffic on the links). As expected, different scaling factors have little effect on the traffic distribution.

[3]The campus network contains 23 links, two of which do not have background traffic, because they are not traversed by flows generated between the access routers based on shortest paths.

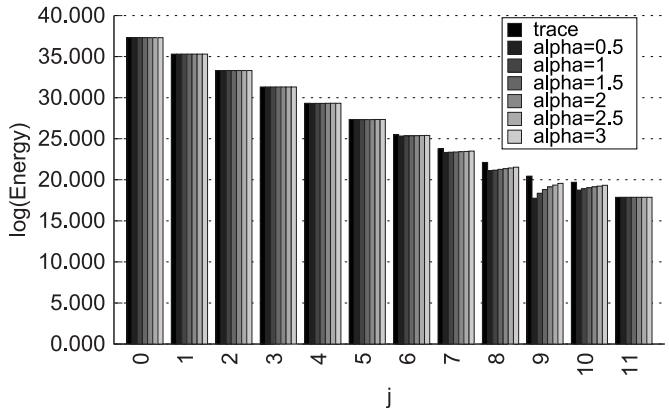

Fig. 9. Energy plot of the CAMPUS trace and generated traffic on one link.

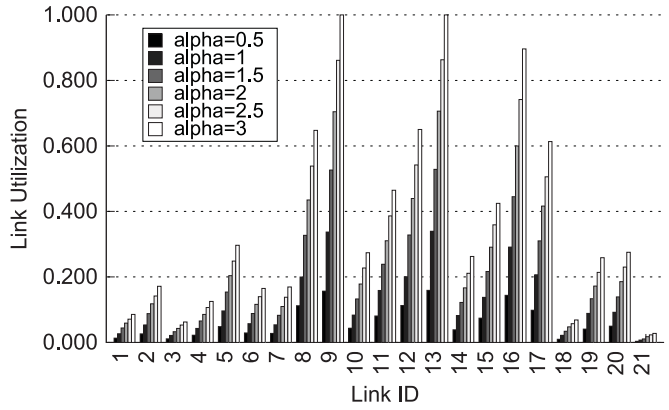

Fig. 10. Link utilization on the campus network with different scaling factors.

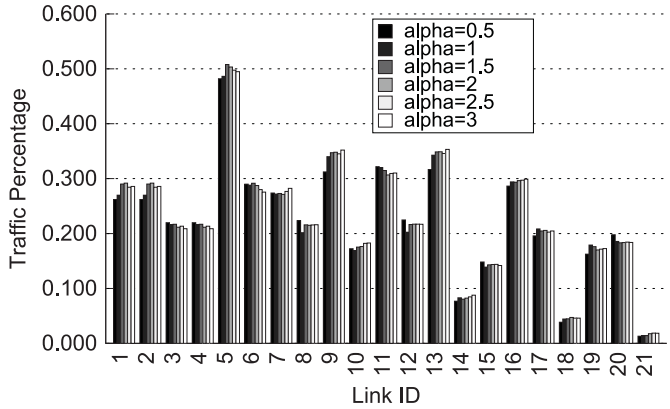

Fig. 11. Traffic distribution on the campus network with different scaling factors.

Table IV. Link Utilization and Traffic Distribution Summary

	Link utilization		Traffic distribution	
α	Mean	Stdev	Mean	Stdev
0.5	0.0565	0.0498	0.2043	0.1164
1	0.1129	0.1019	0.2062	0.1190
1.5	0.1760	0.1589	0.2106	0.1228
2	0.2350	0.2127	0.2099	0.1217
2.5	0.2895	0.2608	0.2093	0.1201
3	0.3501	0.3178	0.2099	0.1207

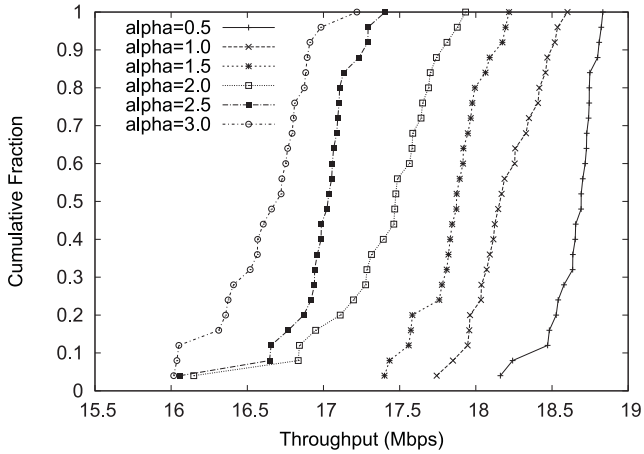

Fig. 12. CDF of TCP throughput with different scaling factors.

Table IV summarizes the results by showing the mean and standard deviation of the link utilization and the traffic distribution.

In the last experiment, we study the effect of the generated background traffic on the behavior of foreground traffic. In a previous study, Vishwanath and Vahdat [2008] showed that background traffic can have significant impact on applications, including web downloads, multimedia video streaming, and bandwidth estimation tools. Here, we simply perform a simulation experiment to show that our generated background traffic can significantly influence the foreground applications. We select two routers in the campus network (one in the 50Mbps network and another in the 20Mbps network) and have them transfer a 100MB file using TCP. We repeat the experiment 25 times for each scaling factor. Figure 12 shows the cumulative distribution function of the measured throughput. As expected, we see that the throughput decreases when the background traffic intensity increases with a larger scaling factor.

8. CONCLUSION AND FUTURE WORK

In this article, we propose a method for generating the background traffic workload that can capture the same spatial and temporal characteristics as observed from the Internet traffic measurement. Our method first classifies traffic according to multidimensional features and then maps the traffic classes onto a given network topology for traffic generation. We validate our simulation method both on a realistic backbone network model and on a synthetic campus network model. The results show that the model is able to generate representative background traffic with important spatial and temporal characteristics.

For future work, we would like to explore in several directions. First, our traffic generation method is based on network measurements, assuming that the user behavior

observed from the packet trace is representative and can reveal the network-wide traffic pattern. It is not enough, however, that one simply relies on a single packet trace; instead, we need to make use of network measurements at different vantage points (e.g., from multiple links of different types at different locations in the network) in order to provide a broader view of the overall network traffic. One may need to judiciously select the traces for a representative global network traffic scenario. We need to identify and deal with the correlations among the different traces. It is also important to represent the idiosyncratic nature of different links at different locations for more realistic network-wide background traffic generation. Our current method relies on the stable behavior of the traffic trace. In order to capture the variance at large timescales (such as the diurnal effect), one may need to extend our method to use traces collected at different time periods to dictate traffic generation for the corresponding time intervals.

Second, our spatiotemporal method can be extended for content-based traffic generation. There are two possibilities. One is to simply extend the traffic generation mechanism (the last step) to produce traffic flows with specific content. In this case, the workload can still preserve the spatiotemporal structure; however, the content may not. The other method is to consider adding features of content when performing traffic classification. In this case, both workload and content can be spatiotemporally correlated. Content-based traffic generation needs to consider the specific type of applications. We would like to explore content-based background traffic generation for important applications, such as network caching, content distribution, and intrusion detection.

Third, we would like to extend the method and introduce more "control knobs" in the model so that the users can easily tune the parameters to generate different traffic scenarios, while maintaining the important spatial and temporal traffic characteristics. There are several possible places one can insert such control knobs. For example, we have introduced a scaling factor for varying the generated background traffic intensity. However, the traffic load is perfectly balanced among all the links in the network. To tip the balance, one can introduce a traffic skew factor in the specification of the optimization problem for mapping the clusters to routers. One can also introduce a similar mechanism when estimating the traffic matrix.

Last, background traffic generation can be computationally expensive for packet-oriented simulation. We need to consider high-performance methods especially for large-scale networks with high-capacity links. We can use fluid models to represent background traffic. We applied a simple fluid model for validation in Section 7.1 in this work, since the experiment does not include interaction with foreground traffic. We will investigate hybrid models using our traffic generation method and integrating fluid-based background traffic and packet-based foreground traffic in future work. It will be a performance—accuracy tradeoff—one needs to accurately and yet efficiently capture the mutual interaction between the foreground applications and background traffic at the proper timescale.

ACKNOWLEDGMENTS

Special thanks to Dr. Tao Li and Dr. Shaolei Ren at Florida International University, who provided help and guidance in the discussions on some of the important technical issues presented in the article. We would also like to thank the anonymous reviewers for their constructive comments and suggestions.

REFERENCES

P. Abry and D. Veitch. 1998. Wavelet analysis of long-range dependent network traffic. *IEEE Transactions on Information Theory* 44, 1 (1998), 2–15.

J. S. Ahn and P. B. Danzig. 1996. Packet network simulation: Speedup and accuracy versus timing granularity. *IEEE/ACM TON* 4 (1996), 743–757.

A. Anand, V. Sekar, and A. Akella. 2009. SmartRE: An architecture for coordinated network-wide redundancy elimination. In *Proceedings of SIGCOMM'09*. 87–98.

P. Barford and M. Crovella. 1998. Generating representative web workloads for network and server performance. In *Proceedings of SIGMETRICS'98*, Vol. 26. 151–160.

S. Bhandarkar, A. L. N. Reddy, Y. Zhang, and D. Loguinov. 2007. Emulating AQM from end hosts. In *Proceedings of SIGCOMM'07*. 349–360.

V. Bharti, P. Kankar, L. Setia, G. Gursun, A. Lakhina, and M. Crovella. 2010. Inferring invisible traffic. In *Proceedings of Co-NEXT'10*. 1–12.

CAIDA. 2011. CAIDA Internet Traces. Retrieved from http://www.caida.org/data/passive/passive_2011_dataset.xml.

A. Dainotti, A. Pescapè, P. Salvo Rossi, F. Palmieri, and G. Ventre. 2008. Internet traffic modeling by means of hidden markov models. *Computer Networks: The International Journal of Computer and Telecommunications Networking* 52, 14 (2008), 2645–2662.

A. Dainotti, A. Pescapè, and G. Ventre. 2009. A cascade architecture for DoS attacks detection based on the wavelet transform. *Journal of Computer Security* 17 (2009), 945–968.

M. A. Erazo and J. Liu. 2013. Leveraging symbiotic relationship between simulation and emulation for scalable network experimentation. In *Proceedings of PADS'13*. 79–90.

V. Erramilli, M. Crovella, and N. Taft. 2006. An independent-connection model for traffic matrices. In *Proceedings of IMC'06*. 251–256.

A. Feldmann, A. Gilbert, P. Huang, and W. Willinger. 1999. Dynamics of IP traffic: A study of the role of variability and the impact of control. In *Proceedings of SIGCOMM'99*. 301–313.

Y. Guo, W. Gong, and D. Towsley. 2000. Time-stepped hybrid simulation (TSHS) for large scale networks. In *Proceedings of INFOCOM'00*, Vol. 2. 441–450.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explorations* 11 (2009), 10–18.

D. Han, R. Grandl, A. Akella, and S. Seshan. 2013. FCP: A flexible transport framework for accommodating diversity. In *Proceedings of SIGCOMM'13*. 135–146.

T. Karagiannis, K. Papagiannaki, and M. Faloutsos. 2005. BLINC: Multilevel traffic classification in the dark. In *Proceedings of SIGCOMM'05*. 229–240.

G. Kesidis, A. Singh, D. Cheung, and W. W. Kwok. 1996. Feasibility of fluid-driven simulation for ATM network. In *Proceedings of GLOBECOM'96*, Vol. 3. 2013–2017.

C. Kim, M. Caesar, and J. Rexford. 2008. Floodless in Seattle: A scalable ethernet architecture for large enterprises. In *Proceedings of SIGCOMM'08*. 3–14.

N. Kothari, R. Mahajan, T. Millstein, R. Govindan, and M. Musuvathi. 2011. Finding protocol manipulation attacks. In *Proceedings of SIGCOMM'11*. 26–37.

A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft. 2004. Structural analysis of network traffic flows. In *Proceedings of SIGMETRICS'04*. 61–72.

M. Lee, N. Duffield, and R. R. Kompella. 2010. Not all microseconds are equal: Fine-grained per-flow measurements with reference latency interpolation. In *Proceedings of SIGCOMM'10*. 27–38.

T. Li, N. Van Vorst, and J. Liu. 2013. A rate-based TCP traffic model to accelerate network simulation. *Transactions of the Society for Modeling and Simulation International* 89, 4 (2013), 466–480.

B. Liu, D. R. Figueiredo, Y. Guo, J. F. Kurose, and D. Towsley. 2001. A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation. In *Proceedings of INFOCOM'01*, Vol. 3. 1244–1253.

X. Liu, X. Yang, and Y. Xia. 2010. NetFence: Preventing internet denial of service from inside out. In *Proceedings of SIGCOMM'10*. 255–266.

J. B. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1. 281–297.

MAWI. 2013. MAWI Traffic Archive. Retrieved http://tracer.csl.sony.co.jp/mawi/.

A. McGregor, M. Hall, P. Lorier, J. Brunskill, A. Mcgregor, M. Hall, P. Lorier, and J. Brunskill. 2004. Flow clustering using machine learning techniques. In *Proceedings of PAM'04*. 205–214.

A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. 2002. Traffic matrix estimation: Existing techniques and new directions. In *Proceedings of SIGCOMM'02*. 161–174.

V. Misra, W. Gong, and D. Towsley. 2000. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proceedings of SIGCOMM'00*. 151–160.

V. Misra, W. Gong, and D. Towsley. 2003. Fluid models and solutions for large-scale IP networks. In *Proceedings of SIGMETRICS'03*. 91–101.

D. M. Nicol. 2001. Discrete event fluid modeling of TCP. In *Proceedings of n WSC'01*.

D. M. Nicol and G. Yan. 2004. Discrete event fluid modeling of background TCP traffic. *TOMACS* 14, 3 (2004), 211–250.

A. Nucci, A. Sridharan, and N. Taft. 2005. The problem of synthetically generating IP traffic matrices: Initial recommendations. *SIGCOMM Computer Communication Review* 35, 3 (2005), 19–32.

P. Papageorge, J. Mccann, and M. Hicks. 2009. Passive aggressive measurement with MGRP. In *Proceedings of SIGCOMM'09*. 255–266.

K. Papagiannaki, N. Taft, Z. Zhang, and C. Diot. 2003. Long-term forecasting of internet backbone traffic: Observations and initial models. In *Proceedings of INFOCOM'03*. 1178–1188.

M. Podlesny and S. Gorinsky. 2008. RD network services: Differentiation through performance incentives. In *Proceedings of SIGCOMM'08*. 255–266.

M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang. 2002. Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning. In *Proceedings of IMW'02*. 91–92.

M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. 2004. Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification. In *Proceedings of IMC'04*. 135–148.

J. Sommers and P. Barford. 2004. Self-configuring network traffic generation. In *Proceedings of IMC'04*. 68–81.

J. Sommers, R. A. Bowden, B. Eriksson, P. Barford, M. Roughan, and N. G. Duffield. 2011. Efficient network-wide flow record generation. In *Proceedings of INFOCOM'11*.

N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. 2004. Measuring ISP topologies with Rocketfuel. *IEEE/ACM TON* 12, 1 (2004), 2–16.

P. Tune and M. Roughan. 2003. Internet traffic matrices: A primer. *ACM SIGCOMM eBook: Recent Advances in Networking* (2003).

V. C. Valgenti and M. S. Kim. 2012. An application-level content generative model for network applications. In *Proceedings of SIMUTools'02*. 47–56.

N. Vegh. 2013. NTools traffic generator/analyzer and network emulator package. http://norvegh.com/ntools. (2013).

K. V. Vishwanath and A. Vahdat. 2006. Realistic and responsive network traffic generation. In *Proceedings of SIGCOMM'06*. 111–122.

K. V. Vishwanath and A. Vahdat. 2008. Evaluating distributed systems: Does background traffic matter. In *Proceedings of the 2008 USENIX Technical Conference*. 227–240.

Y. Wang, H. Wang, A. Mahimkar, R. Alimi, Y. Zhang, L. Qiu, and Y. R. Yang. 2010. R3: Resilient routing reconfiguration. In *Proceedings of SIGCOMM'10*. 291–302.

S. Wei, J. Mirkovic, and E. Kissel. 2006. Profiling and clustering internet hosts. In *DMIN*. 269–275.

M. C. Weigle, P. Adurthi, F. Hernández-Campos, K. Jeffay, and F. D. Smith. 2006. Tmix: A tool for generating realistic TCP application workloads in ns-2. *CCR* 36, 3 (2006), 67–76.

K. Xu, Z. Zhang, and S. Bhattacharyya. 2005. Profiling Internet backbone traffic: Behavior models and applications. In *Proceedings of SIGCOMM'05*. 169–180.

Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. 2005a. Network anomography. In *Proceedings of IMC'05*. 317–330.

Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. 2003a. Fast accurate computation of large-scale IP traffic matrices from link loads. In *Proceedings of SIGMETRICS'03*. 206–217.

Y. Zhang, M. Roughan, C. Lund, and D. Donoho. 2003b. An information-theoretic approach to traffic matrix estimation. In *Proceedings of SIGCOMM'03*. 301–312.

Y. Zhang, M. Roughan, C. Lund, and D. Donoho. 2005b. Estimating point-to-point and point-to-multipoint traffic matrix: An information-theoretic approach. *IEEE/ACM TON* 13 (2005), 947–960.

Y. Zhang, M. Roughan, W. Willinger, and L. Qui. 2009. Spatio-temporal compressive sensing and internet traffic matrices. In *Proceedings of SIGCOMM'09*. 267–278.

Received January 2014; revised May 2014; accepted August 2014