Florida International University
FIU Digital Commons

FIU Electronic Theses and Dissertations | University Graduate School

2-28-2014

Background Traffic Modeling for Large-Scale Network Simulation

Ting Li
Florida International University, tli001@fiu.edu

DOI: 10.25148/etd.FI14040803
Follow this and additional works at: https://digitalcommons.fiu.edu/etd

 Part of the Other Computer Sciences Commons

Recommended Citation
Li, Ting, "Background Traffic Modeling for Large-Scale Network Simulation" (2014). *FIU Electronic Theses and Dissertations*. 1242.
https://digitalcommons.fiu.edu/etd/1242

This work is brought to you for free and open access by the University Graduate School at FIU Digital Commons. It has been accepted for inclusion in FIU Electronic Theses and Dissertations by an authorized administrator of FIU Digital Commons. For more information, please contact dcc@fiu.edu.

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

BACKGROUND TRAFFIC MODELING FOR LARGE-SCALE NETWORK SIMULATION

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Ting Li

2014

To: Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Ting Li, and entitled Background Traffic Modeling for Large-Scale Network Simulation, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Giri Narasimhan

Tao Li

Shaolei Ren

Gang Quan

Jason Liu, Major Professor

Date of Defense: February 28, 2014

The dissertation of Ting Li is approved.

Dean Amir Mirmiran
College of Engineering and Computing

Dean Lakshmi N. Reddi
University Graduate School

Florida International University, 2014

© Copyright 2014 by Ting Li

All rights reserved.

ACKNOWLEDGMENTS

I would first like to express my sincere gratitude to my advisor, Dr. Jason Liu, for his help and patience over the last six years. He continuously provided the vision, advice and support necessary for me to proceed through the doctorial program and complete my dissertation.

I want to thank members of my committee. Special thanks go to Dr. Giri Narasimhan, Dr. Tao Li, Dr. Shaolei Ren, and Dr. Gang Quan, for their warm encouragement, constructive comments and insightful suggestions. I should also mention Dr. Jeffrey Fan, who used to be in my committee. I am grateful for the assistance given by him.

I further wish to thank my constant collaborators, Nathanael Van Vorst and Miguel Erazo, for our illuminating discussions, for all the best and worst moments we experienced together. Your friendship and generous assistance meant more to me than I could ever express. I am also deeply grateful to work in a vibrant and cheerful group. I would like to thank Yue Li, Ying He, Rong Rong, and Hao Jiang. A special thanks goes to my friend Lixi Wang. Without our shared time, I would feel extremely lonely during this journey.

My deepest appreciation goes to my parents Zhixun Li and Mingming Li. Their boundless love and encouragement offered me invaluable spiritual support. The last two people I want to thank are my husband Tieyong Hu and my son Colin. It is my little boy who reminds me daily to be a wonderful role model to him. His bright smile is an inexhaustible source of laughter, joy and energy. As for my husband, I owe him everything and I wish I could express how much I love and appreciate him. He is a devoted husband and father; he is my best friend. Without his love, support and sunny optimism, this dissertation would not have been possible.

ABSTRACT OF THE DISSERTATION

BACKGROUND TRAFFIC MODELING FOR LARGE-SCALE NETWORK SIMULATION

by

Ting Li

Florida International University, 2014

Miami, Florida

Professor Jason Liu, Major Professor

Network simulation is an indispensable tool for studying Internet-scale networks due to the heterogeneous structure, immense size and changing properties. It is crucial for network simulators to generate representative traffic, which is necessary for effectively evaluating next-generation network protocols and applications. With network simulation, we can make a distinction between foreground traffic, which is generated by the target applications the researchers intend to study and therefore must be simulated with high fidelity, and background traffic, which represents the network traffic that is generated by other applications and does not require significant accuracy. The background traffic has a significant impact on the foreground traffic, since it competes with the foreground traffic for network resources and therefore can drastically affect the behavior of the applications that produce the foreground traffic. This dissertation aims to provide a solution to meaningfully generate background traffic in three aspects. First is realism. Realistic traffic characterization plays an important role in determining the correct outcome of the simulation studies. This work starts from enhancing an existing fluid background traffic model by removing its two unrealistic assumptions. The improved model can correctly reflect the network conditions in the reverse direction of the data traffic and can reproduce the traffic burstiness observed from measurements. Second is scalability. The trade-off between accuracy and scalability is a constant theme in background traffic modeling. This

work presents a fast rate-based TCP (RTCP) traffic model, which originally used analytical models to represent TCP congestion control behavior. This model outperforms other existing traffic models in that it can correctly capture the overall TCP behavior and achieve a speedup of more than two orders of magnitude over the corresponding packet-oriented simulation. Third is network-wide traffic generation. Regardless of how detailed or scalable the models are, they mainly focus on how to generate traffic on one single link, which cannot be extended easily to studies of more complicated network scenarios. This work presents a cluster-based spatio-temporal background traffic generation model that considers spatial and temporal traffic characteristics as well as their correlations. The resulting model can be used effectively for the evaluation work in network studies.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Network simulation is an indispensable tool for studying Internet-scale networks due to the heterogeneous structure, immense size and changing properties. It is crucial for network simulators to generate representative traffic, which is necessary for effectively evaluating next-generation network protocols and applications. With network simulation, we can make a distinction between *foreground traffic*, which is generated by the target applications the researchers intend to study and therefore must be simulated with high fidelity, and *background traffic*, which represents the bulk of network traffic generated by other applications and does not require significant accuracy. The background traffic has a significant impact on the foreground traffic, since it competes with the foreground traffic for network resources and therefore can drastically affect the behavior of the applications that produce the foreground traffic. This dissertation aims to provide a solution to meaningfully generate background traffic in three aspects. The first aspect is realism. Realistic traffic characterization plays an important role in determining the correct outcome of the simulation studies. This work starts from enhancing an existing fluid background traffic model by removing two unrealistic assumptions. The second aspect is scalability. The trade-off between accuracy and scalability is a constant theme in background traffic modeling. This work presents a fast rate-based TCP traffic model, which outperforms other existing traffic models in that it can correctly capture the overall TCP behavior and achieve a speedup of more than two orders of magnitude over corresponding packet-oriented simulation. The third aspect is network-wide traffic generation. Regardless of how detailed or scalable the existing models are, they mainly focus on how to generate traffic on one single link, which cannot be extended easily to studies of more complicated network scenarios. This work presents a cluster-based spatio-temporal background traffic generation model that considers spatial and temporal traffic characteristics as well as their

correlations. The resulting model can be applied to the entire network for evaluating new applications and protocols under complicated scenarios in network studies.

1.1 Motivation

As the size of Internet increases, efficiently studying the behavior of the Internet-scale networks becomes more and more challenging. Because of the complexity of the network behavior there is no available analytical model that can accurately describe the behavior of Internet-scale networks in every aspect. This makes large-scale network simulation an indispensable tool for studying immense networks. Previous work has shown that in certain cases, only large-scale simulation is able to gain credible evidence. Some applications only show full potential on large-scale environments. For example, scalability is critical for peer-to-peer applications since they exhibit "network effect", which means the behavior of one user is affected when another user joins and enlarges the network [RFI02]. Another example is Internet worm study, which requires a large-scale model to show the propagation dynamics of the worm [LYPN02]. However, it is a great challenge to represent the behavior of network applications under a wide variety of large-scale traffic conditions in a repeatable and controllable fashion. Modeling and simulating the Internet-scale networks is a difficult task due to the heterogeneous structure, immense size and changing properties of today's network [FP01].

Topology, traffic, and scenario are three important factors for network simulation. A fundamental component of network simulation is traffic modeling, which is essential for evaluating next-generation network applications and protocols. Floyd and Kohler have strongly advocated the use of better models for network research through careful examination of unrealistic assumptions in modeling and simulation studies [FK03]. More realistic models often imply more costly simulations in terms of time and space. This

work focuses on addressing the challenges facing the background traffic modeling for large-scale network simulation.

There have been numerous attempts to model the background traffic for large-scale network study. Unfortunately, there are important issues left to be addressed for modeling and simulating background traffic for large-scale network simulation. First, it is critical to preserve the important realistic characteristics of the real network traffic in the background traffic models for network experimentation. Floyd and Kohler [FK03] showed through a series of counter examples that realistic traffic characterization (including, for example, proper traffic load distribution and bidirectional traffic) plays an important role in determining the correct outcome of the simulation studies. The same observation has been confirmed in the performance studies of high-speed TCP protocols. Ha et al. [HLRX07] demonstrated conclusively that the stability, fairness, and convergence speed of several TCP variants are clearly affected by the intensity and variability of background traffic. In addition, Vishwanath and Vahdat conducted a more systematic study on the impact of background traffic on distributed systems, including Web applications, multimedia video streaming applications, and bandwidth estimation tools [VV08]. They concluded that even small differences in the burstiness of background traffic can lead to drastic changes in the overall application behavior. Therefore, how to preserve the essential characteristics of the real traffic is one main challenge of modeling the background traffic.

There have been considerable efforts in generating repeatable and realistic traffic to support performance evaluation of network applications and distributed systems for both simulation and emulation. Notable examples of existing traffic generators include Surge [BC98], Harpoon [SB04], Tmix [WAHC$^+$06], and Swing [VV06]. These traffic models are all packet-oriented and represent the background traffic in detail. Each packet-level instance such as packet arrival or packet departure is processed as a simu-

lation event. The computational cost increases proportionally as the number of packets increases, which results in an inintolerable computational burden for large-scale network simulation. Scalability therefore becomes the second challenge to model the background traffic.

To reduce the simulation complexity and model the background traffic efficiently, researchers resort to abstracted fluid models (e.g., [AD96], [Nic01], [LPM$^+$03]). In fluid simulation, network traffic is modeled in terms of continuous fluid flows rather than individual packet instances. In addition to pure packet models and pure fluid models, there are existing hybrid models (e.g., [GGT00], [RJF02], [KSWU03], [NY04], [GLT04], [Liu06]) aimed at capturing the interactions between fluid background traffic and foreground packet-oriented flows. These hybrid models can achieve substantial performance gains—in some cases, more than three orders of magnitude—over traditional packet-oriented simulation, while still maintaining good accuracy. One reason for the existence of many different background traffic models is that people have realized that one single model is not able to satisfy the requirement of various network studies. The type of background traffic model to be used depends on the objective of the specific network study. Therefore, it is necessary to provide a set of background traffic models with different levels of accuracy and efficiency.

Regardless of whether the emphasis of the traffic model is accuracy or performance, earlier work has mainly focused on how to model realistic and scalable background traffic on one single link. The results are a temporal series of packets or flows arrivals and departures without any spatial correlations. It cannot be extended easily to the studies of more complicated network scenarios. Therefore, a spatio-temporal background traffic model which can be applied network-wide is needed. Many studies have been done on the spatio-temporal analysis of the network traffic, such as [FTT02], [YM05] and [WAF99]. However, to the best of our knowledge, there has been little work on spatio-temporal

background traffic generation for Internet-scale network simulation. In principle, we can repeat the background traffic generation on one link for all the links of the network until we reproduce the network-wide traffic. Unfortunately this is not feasible for most cases because measured data for all the links may not be available. Even if we assume a complete set of link observations, given the immense size and the complexity of the large-scale network, it would be extremely time consuming and difficult to manage such a great amount of data and reproduce the traffic link by link based on the characteristics derived from the measured data. Therefore, some simplification must be made to reduce the complexity of background traffic generation for large-scale network model.

In this dissertation we make a thorough study of the existing background traffic models and related work, and we present our solutions of modeling background traffic for large-scale network simulation in terms of improving its accuracy, scalability and incorporating both temporal and spatial traffic characteristics.

1.2 Problem Definition

Background traffic modeling is a key component of network simulation. In this section we formalize three problems regarding background traffic modeling we intend to address in this dissertation.

- **Realistic Background Traffic Modeling**: In most cases the evaluations of the applications and protocols are only valid when applying realistic background traffic in the network experimentation. There have been numerous attempts to model the background traffic with the goal of realism (e.g., packet-oriented models). The cost of faithfully representing background traffic is substantial, especially for large-scale simulation. Our goal is to generate a realistic and responsive background traffic that preserves the important characteristics observed from the real network traffic with-

out describing it in packet-level details. Our solution is to improve an existing fluid (hybrid) model by eliminating some of the unrealistic assumptions of the model while retaining its computational efficiency.

- **Scalable Background Traffic Modeling**: Traditional packet-oriented simulation typically requires several simulation events to represent a packet visiting each router along its way from the traffic source to the destination (with at least two events to represent the packet arrival and departure at each router). Such detailed packet-level representation of traffic can incur substantial cost when simulating traffic for large-scale networks with millions of hosts and routers [FPP+03]. Furthermore, since the space complexity of a detailed packet-level simulation is proportional to the number of packets stored in the network queues, for networks that consist of links with a large delay bandwidth product, the memory cost can be exorbitant. Existing fluid models address the scalability issue by describing the traffic in the unit of flows instead of individual packets. However, some fluid models only model the long-term average behavior of traffic flows to achieve better performance; other fluid models preserve more detailed information for better accuracy, and consequently, they are not efficient. Therefore, there is need for a new background traffic model which considers both performance and accuracy.

- **Spatio-Temporal Background Traffic Modeling**: We can divide the existing traffic models into spatial and temporal models. *Spatial models* distribute traffic based on traffic matrices. They focus specifically on aggregate traffic intensity (rather than individual flows and packets) and can only deal with variations at coarse time scales (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 and flows. *Temporal models* are based on the traffic traces collected from individual links; they generate traffic as individual flows or packets, and therefore can capture

the effect on the target applications more accurately. Almost all existing traffic models, either the detailed packet-level models or the abstract fluid models, focus on generating the traffic on a single link. They can capture the temporal behavior of the traffic, such as traffic arrival, departure, etc. However, due to the lack of proper spatio-temporal traffic models, there is no agreement or guidelines on how to place the background traffic in complex network environments. Our goal is to provide a spatio-temporal background traffic model, considering both temporal and spatial structures and their correlations when propagating the generated traffic flows on the entire network.

The overarching goal of our work is to create proper background traffic generation models so that the synthetic traffic preserves the important characteristics of the real network, is scalable, and can be applied to the entire network.

1.3 Contributions

In this section we list the major contributions of this work.

1. We propose a fluid background traffic model that can capture the workload characteristics of the Internet traffic for a single link. This model is extended from our previous hybrid model that integrates a fluid-based analytical model using ordinary differential equations (ODEs) with the traditional packet-oriented discrete-event simulation [Liu06]. In particular, we removed two unrealistic assumptions from the previous approach to properly model Internet background traffic. Our experiments show that the fluid background traffic model can provide a realistic representation of the Internet traffic and can accurately capture the interactions with foreground traffic produced by common applications.

2. We propose a fast rate-based TCP model to approximate the behavior of TCP traffic in network simulation at a new level of granularity. In this model, we apply analytical models to capture the approximate behavior of the TCP end systems to further speed up network simulation. We conduct extensive experiments to validate our model and show that it can lead to more than two orders of magnitude reduction both in the number of simulation events and the simulation time for congested networks.

3. We propose a cluster-based spatio-temporal background traffic generation model that can capture both 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 multi-dimensional attributes and group them into distinct classes. We 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. The generator is also capable of scaling the traffic so that the traffic intensity can be varied accordingly in order to test applications under different network conditions. The scaled traffic is shown to maintain the same spatial and temporal characteristics as in the observed traffic traces. This model can be applied to the entire network for evaluating new applications and protocols under complicated scenarios.

This dissertation work is drawn from the following papers:

- A Fluid Background Traffic Model, T. Li and J. Liu, IEEE International Conference on Communications, Page 1-6, June 2009.
- A Rate-Based TCP Traffic Model to Accelerate Network Simulation, T. Li, N. Van Vorst and J. Liu, Transactions of the Society for Modeling and Simulation International, Volume 89, Issue 4, Pages 466-480, April 2013.

- Cluster-Based Spatio-Temporal Background Traffic Generator for Network Simulation, T. Li and J. Liu, submitted to ACM Transactions on Modeling and Computer Simulation (TOMACS).

1.4 Evaluation Criteria

In order for us to study background traffic models, we summarize the following criteria in our study to evaluate the models with respect to their accuracy, efficiency, and flexibility.

- To evaluate the accuracy, we compare the proposed background traffic models with the real measurements or detailed packet-oriented models. To investigate the temporal structures of the generated traffic, we look at the traffic intensity, the number of sessions over time, and the traffic burstiness with different time scales. To investigate the spatial structures of the generated traffic, we compare the generated traffic matrix with the real traffic matrix obtained from measurements. To investigate the correlations between spatial and temporal characteristics of the generated traffic, we study link utilization, traffic distribution, and flow sizes of the traffic observed over all links in the network. To evaluate the effect on the foreground traffic, we investigate the behavior of foreground traffic that is generated by target applications with different background traffic.
- To evaluate the efficiency, we study the reduction in the simulation event count and execution time by running the simulation with the proposed background traffic models and the detailed packet-oriented models under various network scenarios.
- To evaluate the flexibility of the background traffic generation, we tune the traffic model (e.g., vary the traffic intensity by tuning a scaling factor) and study the behavior of the generated traffic.

1.5 Outline

The rest of this dissertation is organized as follows. In Chapter 2, we introduce the existing background traffic models and related work. We review temporal models including packet-oriented, fluid, and hybrid models and discuss the trade-off between accuracy and efficiency. We also review the previous work of spatial models and spatio-temporal models, then present the urgent need of a spatio-temporal network traffic model.

In Chapter 3 we present our work aimed to improve the realism of the existing background traffic model that aims to reproduce network traffic for a single link. In particular, we improved the previous approach [Liu06] to properly model the background traffic by adding two enhancements. The first improvement is to model ACK flows which allows us to correctly reflect the network condition on the reverse direction of the data traffic. The second improvement is to model the traffic sessions by using a PPBP (Poisson Pareto Burst Process) to reflect the self-similarity of the traffic.

In Chapter 4, we describe a fast rate-based TCP traffic model, which is a novel mechanism to model TCP background traffic at a new aggregation level. Rather than modeling the network traffic packet by packet as in the traditional approach, our solution groups the packets with various lengths into rate windows. The model calculates the queuing delays and the packet losses as these rate windows traverse the individual network queues along the flow path. The TCP congestion control behavior is captured using analytical models in response to the simulated network traffic conditions at different phases of a TCP connection. These features of the model accelerate the simulation by more than two orders of magnitude.

In Chapter 5 we present a cluster-based spatio-temporal traffic generator for network-wide background traffic generation. We use clustering techniques to describe the spatio-temporal structures of the traffic as multi-dimensional functions. Then we propose a

method to generate traffic based on the spatio-temporal characteristics to the entire network by formulating and solving the problem as a set of optimization problems.

Finally, in Chapter 6 we present conclusions of our research and discuss future research directions.

CHAPTER 2

BACKGROUND

There have been many attempts to model network background traffic. These models place particular emphasis on different network characteristics. In the following we first review the existing packet-oriented traffic models and fluid traffic models. All these models focus on modeling temporal traffic behaviors in time wise. We then review the related work on the spatial traffic models and the spatio-temporal traffic models.

2.1 Packet-Oriented Traffic Modeling

The transmitted packets in the network are the carrier of the informative network traffic. In traditional packet-oriented network simulation, one or more simulation events are associated with each packet visiting a router (representing both the packet arrival and the packet departure). The packet-oriented models can be used as detailed representations of background traffic. In this section we introduce several methods to model network traffic at packet level.

2.1.1 Traffic Playback

The straightforward and simplest way to regenerate the network traffic is to playback the traffic according to the observed behavior of the packets traversing a real network, reproducing the exact arrival process of the real traffic. Conducting traffic playback of a link requires one or more traces of that particular link. Besides collecting their own traces, researchers can also make use of public repositories of traces, such as the Internet Traffic Archive [Arc], CAIDA [CAIa] and NLANR [NLA].

Although packet-level playback is able to precisely reproduce the traffic of the collected traces, it is not flexible. The researchers have to limit their experiments to the

available traces and the characteristics of the traces. For example, a researcher who intends to evaluate a queuing mechanism under a range of loads must find a collection of traces covering this range of loads using a different trace in each different run of the experiments in order to introduce the variability. In such a way even studying a simple question can be cumbersome. Unfortunately, it is not feasible for most cases because the required data may not be available for reasons such as privacy and security. Thus, the realism of the resulting traffic and the conclusions of the evaluation could be questionable. Additionally, traffic playback is irresponsible to the changes of the network scenarios. Therefore, it is only useful for certain types of experiments where the traffic behavior is not expected to affect the generated traffic. For example, malicious traffic can often be generated by using packet-level playback [SYB04], because malicious traffic (e.g., a SYN flood) is frequently not responsive to network conditions.

Due to the shortcomings of the traffic playback, more flexible traffic generation is needed in network research community. Under this situation, researchers turn to the synthetic traffic generators which generate network traffic by using statistical network characteristics derived from the traces. The challenge then is to determine which properties of traffic are most important to reproduce so that the synthetic traffic behaves in a realistic way. In the following we discuss several measurement-derived traffic generations with respect to different characteristics.

2.1.2 Self-similar or Multi-fractal Traffic Models

In traditional models of network traffic, it is assumed explicitly that traffic characteristics such as packet arrival rate have Poisson distributions. However, Leland et al. [LTWW94] observed that Internet traffic is bursty, and this burstiness can be studied using the framework provided by statistically self-similar process rather than Poisson process. Therefore, when researchers conduct network experiments, in most cases they should consider self-

similarity as an important property of the background traffic applied to the experiments in order to obtain reliable experimental results. It is shown that the bursty traffic observed from Internet can be generated by rejecting packets into the experiments based on a self-similar stochastic process [Pax97b].

Mathematically, statistical self-similarity manifests itself as long-range dependence, a sub-exponential decay of the autocorrelation of a time-series with scale. This is in sharp contrast to Poisson modeling and its short-range dependence, which implies an exponential decay of the autocorrelation with scale. A stochastic process can be called self-similar (with Hurst parameter H) if the rescaled process, with an appropriate rescaling factor, is equal in distribution to the original process [Ber94].

Let $X = \{X_t | t = 0, 1, 2, \ldots\}$ be a wide-sense stationary stochastic process with autocorrelation function $r(k)$, $k \geq 0$. Let $\{X_k^{(m)} | k = 1, 2, 3, \ldots\}$ represent a family of aggregate processes produced by summing the original time series, X, over adjacent, non-overlapping blocks of size m. In particular, for integer $m \geq 1$, $X^{(m)} = \{X_k^{(m)} | k = 1, 2, 3, \ldots\}$, where $X_k^{(m)} = \frac{1}{m} \sum_{i=km-(m-1)}^{km} X_i$. Generally stated, if the distribution of each of the aggregate processes $X^{(m)}$, $m > 1$ is approximately the same as that of the original process, $X = X^{(1)}$, then X is self-similar.

Self-similar processes can usually be described by heavy-tailed distributions, which are also known as long-tailed distributions. A distribution is heavy- tailed if

$$Pr[X > x] \sim x^{-\alpha}, \qquad x \to \infty, \qquad 0 < \alpha < 2. \tag{2.1}$$

This means that regardless of the distribution for small values of the random variable, if the asymptotic shape of the distribution is hyperbolic, it is heavy-tailed. The simplest heavy-tailed distribution is Pareto distribution which is hyperbolic over its entire range. We describe Pareto distribution and how we use it for generating self-similar traffic in Section 3.2 in detail.

Self-similar processes can be classified into two models: One is an exactly self-similar model and the other is an asymptotically self-similar model [Ber94]. For the real applications, an exactly self-similar model is too narrow to model the real traffic, thus in most cases, an asymptotically self-similar model is used. A process is called asymptotically second-order self-similar with parameter $H \in (0.5, 1)$, if its autocorrelation function is with the form

$$r(k) \sim ck^{2H-2}, k \to \infty \tag{2.2}$$

where $c > 0$ is a constant.

So far a lot of works have been done to reproduce self-similarity in simulated networking traffic, and a number of relevant models have been proposed. A survey of models using self-similar stochastic processes is given in [HKS99]. The major models are based on one of the following processes. The first processs is Fractional Brownian Motion (FBM), that is a continuous-time zero mean Gaussian process $B_H(t)$. We can define $B_H(t)$ with Hurst parameter H as follows:

1. $E[B_H(t)] = 0$
2. $B_H(0) = 0$
3. $B_H(t+\delta) - B_H(t)$ is normally distributed $N(0, \sigma|\delta|H)$
4. $B_H(t)$ has independent increments
5. $E[B_H(t)B_H(s)] = \sigma^2/2(|t|^{2H} + |s|^{2H} - |t-s|^{2H})$

FBM is exactly self-similar, which is improper to model the real traffic as we discussed above. However, it can be used to model the sum or integral of self-similar traffic (as observed in network buffers, file sizes of audio/video streams, etc). On the other hand, Fractional Gaussian Noise (FGN) is exactly second-order self-similar traffic and is a good candidate for modeling the traffic of Ethernet, ATM, VBR coded video, Web Telnet

and FTP instances. The increments of FBM are known as Fractional Gaussian Noise (FGN) [MN68] and form a stationary process $G_H(t)$ with the following properties:

1. $G_H(t) = (B_H(t+\delta) - B_H(t))/\delta$
2. $G_H(t)$ is normally distributed $N(0, \sigma|\delta|^{H-1})$
3. $E[G_H(t+\tau)G_H(t)] = \delta^2 H(2H-1)|\tau|^{2H-2}$ for $\tau \gg \delta$

Fractional ARIMA models (ARFIMA or FARIMA) are built on classical ARIMA models and are asymptotically self-similar [LTWW94], [Ber94]. $\{X_n\}_{n=0}^{\infty}$ is called an ARFIMA(p,d,q) process, if $\{\Delta^d X_n\}_{n=0}^{\infty}$ is an ARIMA(p,q) process for some non-integer $d > 0$. Let B be the Backshift-operator $B(X_n) = X_{n-1}$ and Δ^d can be represented by: $\Delta^d = (1-B)^d = \sum_{u=0}^{\infty} \pi_u B^u$, with $\pi_0 = 0$ and $\pi_u = \frac{\Gamma(u-d)}{\Gamma(u+1)\Gamma(-d)} = \Pi_{k=1}^{\infty} \frac{k-1-d}{k}, u = 1,2,\ldots$ They are similar to FGN, yet they are more flexible because they have more parameters. However, due to their computational complexity, it is more difficult to use ARIMA models than FGN models.

In addition to the above stochastic processes, a large number of superimposed heavy-tailed On/Off processes can yield self-similar traffic as well [CB95]. An On/Off process is either in the On or Off state. If On-times and Off-times are drawn from a heavy-tailed distribution like the Pareto distribution with parameters α_1 and α_2, then the observed stochastic process is self-similar FGN with $H = 3 - \min(\alpha_1, \alpha_2)$. Another set of models that can capture both short-range and long-range dependencies are wavelets. Traffic is generated first in the wavelet domain, and then transformed back into the time domain by applying the inverse wavelet transformation [Fla92], [SLN94]. Hlavacs al et. [HKS99] summarized a set of methods to test self-similarity and the Hurst parameter. The methods include Variance-Time plot, R/S plot, Periodogram, Whittle estimator, correlogram plot and Wavelet estimator.

Modeling the sequence of packets by using stochastic processes, a number of superimposed On/Off processes or wavelets focuses on reproducing the self-similarity of the network traffic. However, the resulting traffic flows are not responsive to the change of network conditions and may not correctly interact with the network applications and protocols under investigation.

2.1.3 Application-specific Traffic Generation

The network traffic varies its behavior according to the application protocols. Therefore, there has been a lot of work that focuses on designing application-specific traffic generators.

Tcplib [DJ91] models the traffic of five different types of Internet applications (FTP, SMTP, NNTP, TELNET and RLOGIN) based on a set of traces collected from UC Berkeley, University of Southern California, and Bell Communication Research. It has two limitations. First, it lacks several important application-specific details. Because this work preceded the growth of the web, Tcplib does not include a model of HTTP traffic, which is critical for today's network traffic simulation. This reflects a major shortcoming of application-specific traffic models: because of the fast development of Internet applications and protocols, researchers need to update the models frequently according to the changes of the applications.

The most frequently referenced models of TCP connection distributions were proposed by Paxson and his colleagues [Pax94], [PF94]. They derived analytical models to describe the the traffic associated with TELNET, NNTP, SMTP and FTP applications based on 3 million TCP connections from a number of wide-area traffic traces and a variety of sources. For example, the analytical models of FTP traffic is shown in table 2.1. However, like Tcplib, the HTTP protocol is also absent from Paxon's models.

Table 2.1: FTP Traffic Models

Variable	**Distribution**	
	Paxon's FTP Model	**Revised FTP Model**
Session Arrivals	Poisson	Poisson
Connection Bytes	Lognormal	Pareto *(upper 15%)* Exponetial *(lower 85%)*
FTP-data Spacing	Lognormal / Log-logistic	Manual / Automated Group
Session Bytes	Lognormal	N/A
Number of Connections per Session	N/A	Pareto

Following the work introduced above, as web traffic plays a more and more important role in today's Internet, several web traffic generators have been proposed. Barford and Crovella [BC98] proposed a web traffic generator called Surge, which can create realistic web traffic workload that matches with empirical measurements, including the distribution of file sizes and the lengths of idle periods between consecutive sessions. Also, Cao et al [CCG$^+$04] perform source-level modeling of HTTP traffic.

Application-specific models are easy to use for generating traffic composed of mixed applications given the proper percentage of each application. However, we need to know that the measurement-derived application-specific models need to be revised regularly. Internet is a changing object - the collected traces are only valid for a particular period of time, previous Internet cannot represent current Internet, and current Internet cannot represent the future Internet. Even for protocols that had been studied, they are still necessary to be revised, especially when it has been long time since the work was first proposed and the applications have changed their behavior dramatically. By using traffic traces captured in February 2003 at University of Central Florida, the researchers became aware that many statistical variables of modern Internet traffic differed from Paxon's distribution models [Pax94], [PF94] and preferred more than before to have heavy-tailed features. Even using heavy-tailed distributions, some variables, such as bytes transferred by FTP, HTTP and SMTP protocols are less straightforward to be modeled. In [LM05], Luo and

Marin revised the Paxon's analytical models and modeled several major Internet applications (FTP, HTTP, SMTP, POP3, SSH) by using mixture distributions. For example, the FTP traffic was revised to be modeled by modified variables with different distributions as shown in Table 2.1.

2.1.4 Capturing Structural Characteristics

Contrary to the application-specific traffic models, application-independent traffic generators generate network traffic at the IP flow level by using a set of distributional parameters that are not specific for any particular application. Harpoon [SB04] models traffic at two levels. The lower level is referred to as the connection level, which describes the size of the file transferred and the inter-connection time. The higher level is referred to as the session level, which considers the number of active sessions and the IP preference in terms of the number of sessions each IP gets involved. Each of these parameters can be specified manually or extracted from packet traces or Netflow data collected at a live router. Although Harpoon considers the IP preference as IP spatial distribution, the method does not concern with the connectivity between the IP addresses. In addition, the temporal characteristics are independent from the spatial distributions. As a result, this method can not capture the correlation between the temporal and spatial structures.

Vishwanath and Vahdat proposed Swing [VV06], a network traffic generator, that can extract salient features from the traffic traces based on a structural model. The characteristics for a structural model can be categorized into three sets: application, communication and network. Application characteristics are determined by the specific application protocol, including the session arrival process, the number of connections, the transportation layer protocols, the distribution of packet sizes, etc. Communication characteristics are determined by the user behavior, including the distribution of the client IPs and the think times between individual requests, etc. The network characteristics are used to describe

the network scenarios, such as link delays, packet loss rates and link capacities. Swing captures the important aspects from the behavior of not only applications but also users and network. The measured data are parsed into four categories: session, RRE (Request-Response-Exchange), connection, and pair. A session initiates when a connection the first time appears with a source IP address. It consists of one or more RREs. An RRE is a series of request and response exchanges, which consists one or more connections. A connection is defined by a pair of source and destination. And a pair means one request and response exchange. Swing divides the traffic generation into two steps. In the first step, Swing parses the important characteristics from the traces into a set of files that describe the empirical distributions of several parameters. These parameters can be grouped into four categories:

- *Session:* client IP, session inter-arrival time, number of RREs.
- *RRE:* RRE inter-arrival time, number of connections.
- *Connection:* server IP, connection inter-arrival time, number of pairs.
- *Pair:* request size, response size, think time (the time interval between requests).

In the second step, the parsed data is used as input for a network emulator ModelNet [VYW+02] to generate traffic that is statistically similar to the original traces.

Packet-oriented models, either simply playing back the traffic traces or modeling the traffic based on the behavior of the applications, capture the details of the network traffic at packet level. A detailed representation of background traffic may incur substantial computational cost. This problem becomes particularly acute in large-scale network simulations. Therefore, scalable solutions are needed to efficiently model background traffic.

2.2 Fluid Traffic Modeling

To characterize large-scale networks efficiently, appropriate scalable models are thus required to capture important characteristics of background traffic. To the best of our knowledge, the concept of fluid model was first proposed in [AMS82] to model data network traffic. In fluid simulation, network traffic is modeled in terms of continuous fluid flows rather than individual packet instances [LFG+01]. For example, a series of packets traversing the virtual network from the same source to the same destination or a cluster of closely-spaced packets can be modeled as a fluid trunk with constant or piecewise linear fluid flow rate. Earlier work (e.g., [AD96], [Nic01], [LPM+03]) has shown that fluid models can be applied to reduce the computational complexity in large-scale network simulation. In this section we introduce the major fluid models used to simulate background network traffic.

2.2.1 Time-Stepping Fluid Models

The fluid models proposed in [MGT00] and [LPM+03] use a set of ordinary differential equations (ODEs) to describe continuous fluid flows and to capture the long-term, average behavior of the persistent TCP flows. In those models the network traffic consists of a number of fluid classes, each being a group of TCP flows originating from the same source to the same destination and thus sharing the same characteristics. One can use a set of time-dependent ordinary differential equations to describe the TCP congestion window size, queue lengths, packet delays as well as losses, and then can solve the set of equations numerically using a fixed step-size Runge-Kutta method. At each time step, the states of the fluid model, e.g., the instantaneous queue lengths are updated by the solutions from the equations.

The limitation of these models is that they assume the fluid flows of one single fluid class have homogeneous behavior and ignore the short-term behavior of the flows. For example, they do not model the slow-start behavior of the TCP flows, instead, they only focus on the long-term and average behavior. For the sake of modeling the short-lived TCP flows which dominate in today's Internet, Marsan et al. [MGG+05] exploited partial differential equations to model TCP mice and elephants in large IP networks. Partial differential equations allow the models to represent the distributions instead of the averages of the characteristics, hence achieving better accuracy in the results. The most attractive property of the fluid models described by a set of ODEs resides in the fact that their complexity depends on the number of differential equations to be solved, which is totally independent of the traffic load (the number of packets).

Baccelli and Hong proposed an alternative fluid model [BH02] to describe the average behavior of the AIMD window for the aggregate TCP flows traversing a network of drop-tail routers. In such a network, the buffer behavior is pulsing, which means the congestions are interleaved to the periods of time where no buffer is overloaded. Therefore, only the window dynamics during the congestion epochs are necessary to be carefully analyzed, and the source sending rate between the congestion epochs can be simply assumed to increase at a constant rate. This allows the development of fluid equations and an efficient methodology to solve them. They also extended their model to represent the interactions of TCP flows by incorporating the short-lived traffic with different RTTs and routes on arbitrary network topologies [BH03]. However, each short-lived traffic flow must be described by two differential equations; one presenting the average window evolution and the other describing the workload evolution. As a consequence, the insensitivity of the complexity with respect to the number of TCP flows is lost.

2.2.2 Discrete Event Fluid Models

In contrast to the continuous fluid models, Nicol [Nic01], [NY04] proposed discrete-event fluid models that use piece-wise constant rate functions to represent TCP flows and use simulation events to represent the changes of the flow rate. This approach can model more closely individual sample paths of TCP traffic, including slow start, congestion avoidance, timeout, data loss, and the fast-retransmit mechanism. The computational saving can be substantial when the flows remain a constant rate for a significant period of time.

However, this method has an important limitation. Its performance is determined by the network conditions. In particular, it can be slow if a substantial number of events are generated. This is the case when the flows experience significant packet loss. In addition, the fluid flows are represented individually, so the memory consumption may limit the number of flows.

2.2.3 Fluid Models for Uncongested Links

For the links that are not congested, Barakat et al. [BTI+02] proposed a traffic model by using a Poisson shot-noise process [DVJ98]. Particularly this model is used for backbone links that are generally over-provisioned because the network is designed so that a backbone link utilization stays below 50% in the absence of link failure [FML+03]. The model relies on the flow-level information observed on an IP backbone link and only intends to capture the dynamics of the traffic at short timescales (i.e., in the order of hundreds of milliseconds). They computed the auto-correlations of the flow inter-arrival times, the flow sizes and the flow durations of the collected traces, and found that these sequences exhibit little correlation due to the uncongestion of the link. Thus the total rate of the traffic flows $R(t)$ on the modeled link at time t, can be modeled as the result of the sum

of the rates of different flows traversing this link, which is described in Equation 2.3.

$$R(t) = \sum_{n \in Z} X_n(t - T_n) \tag{2.3}$$

where T_n is the arrival time of the nth flow, and $X_n(t - T_n)$ is the rate of the nth flow at time t; $X_n(t - T_n)$ is zero when flow n is not active at time t. This model is a Poisson shot-noise process, where shot is synonymous of *flow rate function*. This model is designed to be general without any constraint on the particular flows, the applications, or the protocols, and is simple enough to be used in network operation. However, this model is only suitable for modeling the traffic on uncongested links. It also ignores the traffic behavior at long timescales.

2.2.4 Hybrid Traffic Modeling

In addition, hybrid models (e.g., [GGT00], [RJF02], [KSWU03], [GLT04], [Liu06]) aim at capturing the interactions between fluid background traffic (represented by fluids) and foreground flows (represented as packets). The existing hybrid models can achieve substantial performance gains—in some cases, more than three orders of magnitude—over traditional packet-oriented simulation, while still maintaining good accuracy. They mainly focus on the integration mechanism, the interaction between the foreground traffic and the background traffic, rather than reproducing the important characteristics of the Internet traffic (such as the long-range dependencies confirmed by a large collection of measurement studies).

Fluid models gain efficiency at the cost of accuracy. However, the missing of some important features of the model may result in incorrect results. This dissertation improves an existing hybrid model [Liu06] by removing two unrealistic assumptions to properly reproduce the network behavior. Moreover, this work proposes a new rate-based TCP (RTCP) traffic model, which offers a new level of granularity for simulating TCP traffic.

The model reduces the computation complexity significantly while correctly capturing the overall TCP behavior. Even for congested networks, our model can reduce the amount of simulation events and accelerate simulation by more than two orders of magnitude. For uncongested network conditions, the RTCP model can improve execution time even more drastically. We describe the improved hybrid model in Chapter 3 and the RTCP model in Chapter 4 individually.

2.3 Spatial Traffic Modeling

Regardless of how detailed or scalable the packet-level models and the abstract fluid models are, they focus on reproduce time-related traffic characterstics and can be treated as temporal models. On the other hand, spatial models distribute traffic based on traffic matrices. Traffic matrix represents traffic volume between all origins and destinations in a certain time interval. For network design and network management, such as capacity planning and traffic engineering, traffic matrix is usually required as input for performance evaluation. There has been extensive work on estimating the traffic matrix. The gravity model [ZRDG03] assumes that the traffic between an origin-destination (OD) pair is proportional to the total traffic from the source node to the destination node. The main drawback is that the model assumes independence between the source and destination. To solve this problem, generalized gravity models [ZRLD03, ZRLD05] extend the gravity model by separating traffic into classes and applying the gravity model on each class of traffic. 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 as decision makers decide on the traffic volume and the traffic destination based on user behavior and network configuration [MTS+02]. The independent connections model (ICM) [ECT06] 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. The model assumes independence between the connections. Both DCM and ICM require a large number of parameters to achieve accuracy. The low-rank model [BKS+10] provides a simpler and yet more general model, which can be treated as a weighted sum of gravity models. All above spatial models focus only on the spatial distribution of traffic; the result traffic intensity remains constant during a given time interval, the size of which is usually determined by the measurement cost 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.

2.4 Spatio-Temporal Traffic Modeling

Spatio-temporal models consider both temporal and spatial structures jointly. There are methods for traffic matrix estimation that also focus on time dependent properties. Although we treat them here as spatio-temproal models, they are not really concerned with the spatio-temporal correlations of the traffic. Roughan et al. proposed a temporal model [RGK+02] 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 [TR13] can be used to capture the periodic nature of the OD flows by representing the cyclical signal with a small number of Fourier coefficients. Wavelets [PTZD03, AV98] are also used to capture both short-range and long-range dependencies. Principal components analysis (PCA) [LPC+04] can be used to describe the OD flows by using a small number of "eigenflows". All these methods are data-driven

methods and rely heavily on measurments. The derived temporal characteristics of the traffic is independent of the spatial distribution.

There are two recent papers on the spatio-temporal correlation. Zhang et al. proposed a method [ZRWQ09] 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. proposed an interesting method for network-wide flow record generation [SBE+11]. The method is designed to generate representative benign flows as well as anomalous flows, especially for anomaly detection. It builds on the Harpoon traffic generator [SB04] 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. In particular, the method does not concern with the spatial distribution of the IP addresses and the connectivity between the IP addresses. For example, the method may not reflect the existence of hotspots in the real network. Comparatively, our method takes user behavior, node distribution, network connectivity, and flow-level statistics into consideration. As a result, our method can better capture the spatial and temporal correlations of the traffic over the entire network.

Overall, there is a general lack of available spatio-temporal background traffic models. Consequently, the researchers tend to use simple synthetic traffic models in their evaluation work without considering how it is supposed to be. This dissertation provides a solution to address this issue by describing traffic at cluster-level and generating traffic based on its spatial and temporal stuctures, which is presented in Chapter 5.

CHAPTER 3

A FLUID BACKGROUND TRAFFIC MODEL FOR A SINGLE LINK

In this chapter, we propose a fluid background traffic model that can faithfully reproduce the traffic on a single link and capture the workload characteristics of the Internet traffic. This model is extended from our previous hybrid model that integrates a fluid-based analytical model using ordinary differential equations (ODEs) with the traditional packet-oriented discrete-event simulation [Liu06].

3.1 Introduction

There are many discussions about the impact of background traffic on the network, such as the impact of background traffic on TCP protocols [HLRX07, FW02], the impact of background traffic on Internet applications [PKV07]. Previous studies have shown that the background traffic does affect the entire network environment by competing with the foreground traffic. Recently Vishwanath and Vahdat [VV08] demonstrated that realistic background traffic has a different influence on network than simple synthetic traffic models. Therefore, realistic models of background traffic are inevitably required to provide the meaningful network traffic workloads for the network experiments.

We have reviewed existing background traffic models, including both packet-oriented and fluid models, in Chapter 2. In this chapter we focus on improving the accuracy for fluid background traffic models. In particular, we remove two unrealistic assumptions in the previous approach [Liu06] to properly model Internet background traffic.

The first assumption we remove from the the original hybrid model is that most data traffic is one-way and reverse-path traffic is rarely congested. Under this assumption, the previous model ignores the potentially significant effect of delays and losses that occur to TCP acknowledgments (ACKs). We show through experiments that this assumption is un-

realistic and can sometimes lead to completely wrong results. Consequently, we augment the model with fluid ACK flows on the reverse path corresponding to the data flows on the forward path. In doing so, the packet delays and losses in both directions are recorded properly, which are then used to calculate the correct TCP congestion window behavior. Some of these fluid models consider ACK flows. For example, Guo et al. [GGT00] proposed a time-stepped model in which all packets arriving within a time-step are lumped together as a "chunk". At the flow destination, packets in a chunk are individually acknowledged. Nicol and Yan [NY04] applied simple rate equations to describe the behavior of ACK flows. Compared with our solution, both models require more detailed (and more complex) logic to handle ACKs and would therefore instigate more cost.

The second assumption to remove is that traffic sessions are long-lived and thus can be represented by a fixed number of flows between a source-destination pair. Obviously this contradicts observations from traffic measurement studies: session length is known to obey long-tail distributions. In the new model, we adopt the Poisson Pareto Burst Process (PPBP) model [ZNA03] to describe the number of active flows at any given time—with Poisson arrivals and Pareto-distributed durations. Our experiments show that the fluid background traffic model can provide a realistic representation of the Internet traffic and can accurately capture the interactions with foreground traffic produced by common applications.

3.2 The Model

To represent background traffic, we extend our hybrid model [Liu06] that combines discrete-event packet flows and continuous fluid flows described by a set of ordinary differential equations. The fluid component of this hybrid model is based on the TCP model previously proposed by Misra et al. [MGT03].

The fluid network traffic consists of a number of fluid classes, each being a group of TCP flows originated from the same source to the same destination and thus sharing the same characteristics. One can use a set of time-dependent ordinary differential equations to describe the TCP congestion window size, queue lengths, and packet delays and losses. To properly integrate the fluid model with the foreground packet flows, we consider in these equations the effect of packet flows on the fluid flows at each network queue. We then solve the set of equations numerically using a fixed step-size Runge-Kutta method. At each time step, the solutions from the fluid model, e.g., the instantaneous queue lengths, are used to determine the foreground packet delays and possible packet losses. We refer the readers to the original paper for a detailed description of our hybrid approach [Liu06].

3.2.1 Adding ACK Flows

We assume that the forward path of the TCP flows in the fluid class i (from the source to the destination) consists of n queues: $f_1, f_2, \cdots, f_n$, and the reverse path (from the destination to the source) consists of m queues: $r_1, r_2, \cdots, r_m$. The following equation characterizes the additive increase and multiplicative decrease behavior of the TCP congestion window in the fluid model:

$$\frac{dW_i(t)}{dt} = \frac{1}{R_i(t)} - \frac{W_i(t)}{2} \cdot \lambda_i(t), \tag{3.1}$$

where $W_i(t)$ is the size of congestion window, $R_i(t)$ is the round-trip time, and $\lambda_i(t)$ is the packet loss rate of fluid class i. Naturally, $W_i(t)$ should be limited between zero and the maximum congestion window size.

Given the TCP congestion window size, we can calculate the send rate at the source, denoted by $A_i^{f_1}$, which is the arrival rate of fluid class i at the first queue f_1:

$$A_i^{f_1}(t) = \frac{n_i W_i(t)}{R_i(t)}, \tag{3.2}$$

where n_i is the (constant) number of homogeneous fluid flows in the fluid class i. Note that we can easily model constant-rate UDP flows by simply assigning a constant value to $A_i^{f_1}$.

To determine the round-trip time $R_i(t)$ and the end-to-end packet loss rate $\lambda_i(t)$, we accumulate the packet delays and packet losses along the path of each fluid class. We denote $\delta_l^i(t)$ and $\gamma_l^i(t)$ to be the cumulative delay and the cumulative packet loss of the fluid class i at the last queue r_m at time t. Assuming packet losses are uniformly distributed among the TCP flows in the fluid class, we have:

$$R_i(t) = \delta_{r_m}^i(t) \tag{3.3}$$

$$\lambda_i(t) = \gamma_{r_m}^i(t)/n_i. \tag{3.4}$$

Our fluid background traffic model consider both data and ACK flows. We accumulate the packet delays and losses both on the forward path and on the reverse path. At the first queue $l = f_1$, we have:

$$\delta_l^i(t_f) = q_l(t)/C_l \tag{3.5}$$

$$\gamma_l^i(t_f) = A_i^l(t)p_l(t), \tag{3.6}$$

where $q_l(t)$ is the instantaneous queue length, C_l is the link bandwidth, $t_f = t + q_l(t)/C_l$, and $p_l(t)$ is the packet loss probability at queue l. That is, the accumulative delay at the first queue considers only the queuing delay; the accumulative loss at the first queue is the packet arrival rate (from Equations 3.2) multiplied by the loss probability (which can be calculated from the queue length).

For subsequent queues $l \in \{f_2, \cdots, f_n, r_1, \cdots, r_m\}$, we add the delay and loss at the queue to the corresponding values at the predecessor queue. For accumulative delay we

also need to add the link propagation delay. We have:

$$\delta_l^i(t_f) = \delta_b^i(t-a_b) + a_b + q_l(t)/C_l \tag{3.7}$$

$$\gamma_l^i(t_f) = \gamma_b^i(t-a_b) + A_i^l(t)p_l(t), \tag{3.8}$$

where b is the predecessor queue of l, and a_b is the link propagation delay between queue b and queue l.

The changes to the instantaneous queue length can be determined by the difference between the arrival rate (minus loss) and the departure rate:

$$\frac{dq_l(t)}{dt} = \xi_l(t)(1-p_l(t)) - C_l \tag{3.9}$$

where $\xi_l(t)$ is the aggregate arrival rate of both packet and fluid flows. The packet arrival rate can be calculated, for example, as the number of packets arrived at queue l during each Runge-Kutta time-step divided by the step size. The fluid arrival rate is the sum of the arrival rate of all fluid classes traversing queue l. We should limit the queue length between zero and the maximum buffer size.

Equation (3.2) is used to calculate the arrival rate at the first queue f_1. For subsequent queues except r_1, i.e., $l \in \{f_2, \cdots, f_n, r_2, \cdots, r_m\}$, we calculate the arrival rate from the departure rate D_i^b at the predecessor queue b:

$$A_i^l(t+a_b) = D_i^b(t). \tag{3.10}$$

For queue r_1 (the first queue on the reverse path), we have:

$$A_i^{r_1}(t) = \alpha_i D_i^{f_n}(t)/\beta_i, \tag{3.11}$$

where α_i is the average ACK packet size, and β_i is the average data packet size in fluid class i. This equation represents the conversion from the data flows on the forward path to the corresponding ACK flows on the reverse path, assuming that there is a one-to-one mapping between data and ACK packets. Note that this one-to-one assumption may

not hold when TCP enables delayed ACKs, in which case the ratio can be readjusted to compensate for this effect. We did not observe any significant effect of this assumption in our experiments.

The departure rate at queue l is a function of the arrival rate, the bandwidth, and the packet loss rate:

$$D_i^l(t_f) = \begin{cases} A_i^l(t)(1-p_l(t)) & \text{if } \xi_l(t)(1-p_l(t)) \leq C_l \\ A_i^l(t)C_l/\xi_l(t) & \text{otherwise.} \end{cases} \tag{3.12}$$

The packet loss rate is determined by the queue length. For drop-tail queues, it becomes non-zero when the total arrival rate is larger than the link bandwidth and when the queue is full. The loss rate is the amount of overflow divided by the sample period. For RED queues, the loss rate is determined by the average queue length.

3.2.2 Adding Heavy-Tail Session Lengths

The Poisson Pareto Burst process (PPBP) has been shown to able to accurately predict the aggregate Internet traffic [ZNA03]. Here, we use the PPBP model to describe the lengths of TCP flows in the fluid background traffic model.

PPBP is a process based on multiple overlapping bursts, with a Poisson arrival process and burst lengths following a heavy-tail distribution. In our case, we schedule TCP session arrivals using the exponential distribution with a mean arrival rate μ. The durations of the TCP sessions d are independent and identically distributed Pareto random variables with parameters $\delta > 0$ and $1 < \gamma < 2$:

$$P_r(d > x) = \begin{cases} (x/\delta)^{-\gamma} & \text{if } x \geq \delta \\ 1 & \text{otherwise.} \end{cases} \tag{3.13}$$

The mean of the Pareto distribution is $\frac{\delta\gamma}{\gamma-1}$ and the variance is infinite.

Using the Pareto distributed flow duration, we can regenerate the long range dependence (LRD) characteristic of realistic background traffic in our model, which is evaluated by a parameter called the Hurst parameter:

$$\mathscr{H} = \frac{3-\gamma}{2}. \tag{3.14}$$

When $0.5 < \mathscr{H} < 1$, it implies that the traffic exhibits LRD and is self-similar.

We replace the constant number of homogeneous fluid flows within each fluid class i with the PPBP process, $N_i(t)$. Specifically, we redefine the equations for calculating the arrival rate of fluid class i at the first queue f_1 (Equation 3.2), and the end-to-end packet loss rate (Equation 3.4) as follows:

$$A_i^{f_1}(t) = \frac{N_i(t)W_i(t)}{R_i(t)} \tag{3.15}$$

$$\lambda_i(t) = \gamma_{r_m}^i(t)/N_i(t) \tag{3.16}$$

In the implementation, we let the user choose the Hurst parameter $\mathscr{H}$, which we use to determine the parameter $\gamma = 3 - 2\mathscr{H}$. We also ask the user to choose the average number of TCP sessions $\bar{n}$ and the session arrival rate μ for each fluid class i, which we use to determine the mean session length $\bar{d} = \bar{n}/\mu$. Thus, we can determine the parameter δ:

$$\delta = \frac{\bar{d}(\gamma-1)}{\gamma} = \frac{2\bar{n}(1-\mathscr{H})}{\mu(3-2\mathscr{H})} \tag{3.17}$$

We can therefore track the number of active TCP sessions, $N_i(t)$, t, by scheduling session arrivals using an exponentially distributed inter-arrival time with mean of $1/\mu$, and then scheduling departures using session lengths derived from the Pareto distribution with parameters γ and δ.

3.3 Experiments

In this section we describe the simulation experiments to demonstrate that our fluid background traffic is able to capture realistic Internet traffic behaviors.

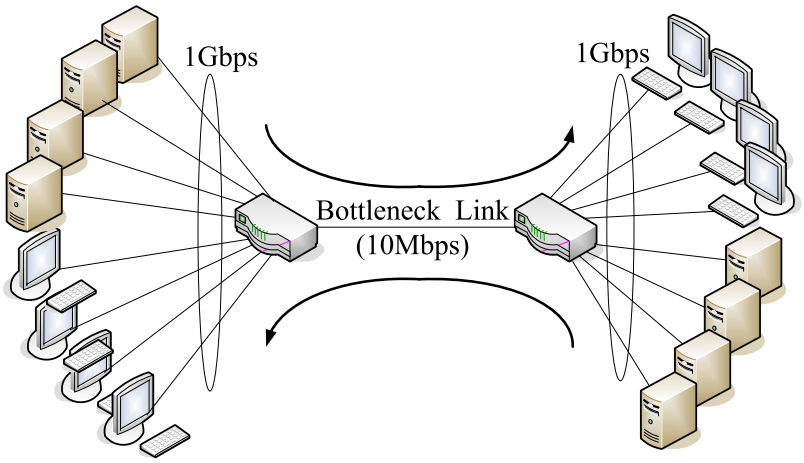


Figure 3.1: A simple dumbbell network

3.3.1 Round-Trip Traffic

We use a dumbbell topology (shown in Figure 3.1) to demonstrate the necessity of including ACK flows in the background traffic model. The dumbbell network model consists of 2 routers and 16 hosts with 4 servers and 4 clients on each side of the dumbbell. The single bottleneck link in the middle has a bandwidth of 10 Mb/s and a link delay of 5 ms. We set the bandwidths of all other links to be 1 Gb/s. The delays of the branch links connecting the end hosts with the router are set so that we have different round trip delays of 10, 20, 50, and 100 ms between a pair of server and client on either side of the dumbbell. The network queues in the routers connecting the bottleneck link each have a 500 KB buffer and are configured with RED parameters q_{min} = 25 KB, q_{max} = 250 KB, and p_{max} = 0.2. The weight used in the EWMA computation for calculating the average queue length is 0.001. We use TCP Reno with the maximum congestion window size of 32.

To emphasize the effect of modeling bidirectional flows, we use a constant number of flows in this experiment. We relax this restriction in the next section to model the long-range dependencies of the traffic. At the start of the simulation, each of the servers on the left side generates 5 TCP flows to the corresponding client on the right side. Between

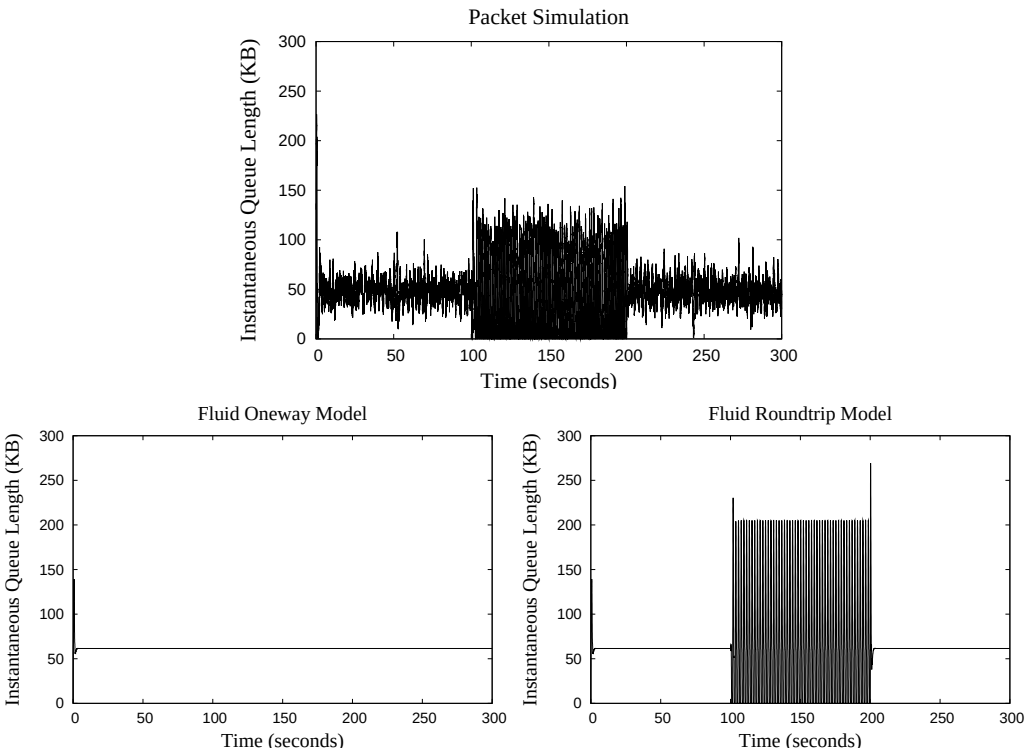


Figure 3.2: Packet vs. fluid one-way and round-trip traffic models

time 100 and 200 seconds, we inject 5 TCP flows in the reverse direction, from each of the four servers on the right side to the corresponding client on the left side. We measure the queue length at the left router connecting to the bottleneck link.

Figure 3.2 shows the results from both packet and fluid simulations. From the packet simulation result (the top plot), we observe a significant variation in the queue length during the period between 100 and 200 seconds when the reverse traffic is introduced. This result is consistent with the conditions observed in the real world, as reported in [SCK+07]. The ACK packets are dropped due to the link congestion caused by the reverse traffic; as a result, it triggers the TCP congestion control mechanism to reduce the sending rate in the forward path. That is, even though the packet loss in the ACK flows should only indicate congestion happens in the reverse path, this indication is interpreted by the source of the forward path (rather than the true source of the congestion at the reverse

path) and causes its TCP to reduce its congestion window. The result is the instability of the queue length during this period.

The bottom plots of Figure 3.2 show the results from fluid simulations. The previous fluid model (i.e., the one-way traffic model) ignores the queuing delays and packet losses in the ACK flows. We can see that the queue length (bottom left of Figure 3.2) remains stable even when the reverse path experiences a heavy congestion, which deviates from reality. In comparison, the round-trip model generates results similar to those from the packet simulation. Although it would be difficult for the fluid model to render exactly the same results as the packet-by-packet simulation, the results from the fluid round-trip model (bottom right of Figure 3.2) clearly indicate the instability of the queue length when the reverse path is congested.

3.3.2 Traffic Burstness

We use the same dumbbell topology as in the previous experiment. We generate one class of 5 TCP flows from each of the four servers on the left of the dumbbell to the corresponding client on the right. In the reverse direction, we generate one TCP flow from each server on the right to the client on the left. That is, the traffic demand from left to right is five times of that from right to left. We select 0.8 as the Hurst parameter and 45 ms as the session inter-arrival time.

Figure 3.3 shows the number of TCP sessions that are active over time from both packet-oriented simulations (left plots) and the fluid background traffic model (right plots). From top down we progressively decreasing the sampling time scale, while maintaining the number of samples to be 300. The starting time scale is 1 second; each subsequent plot is obtained from the previous one by concentrating on a randomly chosen subinterval, the length of which is one tenth of the previous one. That is, the time resolution increases by

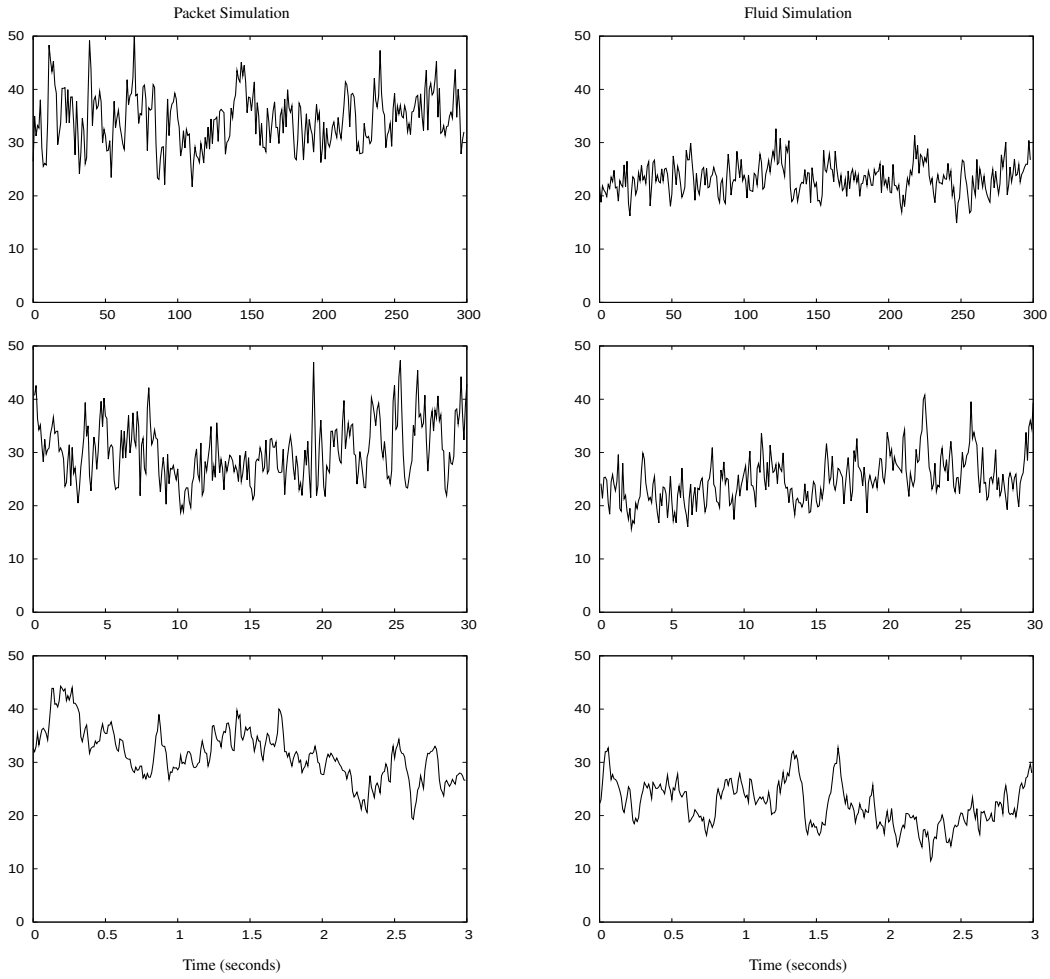


Figure 3.3: Comparison of the number of TCP sessions over time.

a factor of 10. The figure shows both packet simulation and the fluid background traffic model generate similar number of TCP sessions.

We monitor the traffic intensity in the simulations at the same three different time scales as before. For the fluid background traffic model, we calculate the number of packets per second using numerical values at each Runge-Kutta time step, including the congestion window size, round-trip time, and the number of active TCP sessions (Equation 3.15). Figure 3.4 shows the results. As before, the packet simulation results are shown on the left and the fluid simulation results are on the right. The time scale decreases progressively from top to bottom. To a large extent, the results from the packet

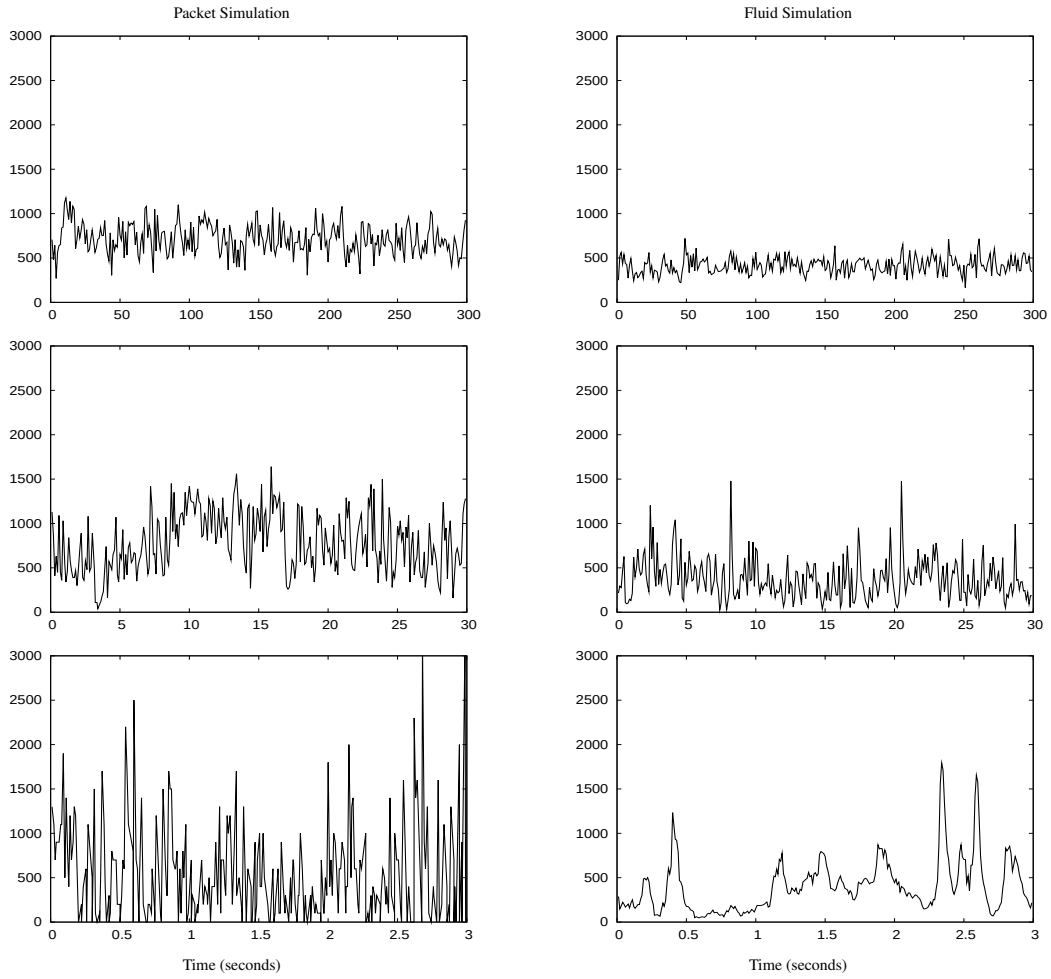


Figure 3.4: Burstness of traffic intensity (packets/second)

simulation and from the fluid simulation are similar, except for the 10 ms timescale (bottom plots). The fluid model does not capture packet details at sub-RTT level; the RTT for the dumbbell model is at least 10 ms.

As before, we monitor the queue length at the left router connecting to the bottleneck links (Figure 3.5). The results from both packet and fluid simulations are similar, although the fluid model creates more variations in the queue lengths. We believe this is partially due to the synchronization effect: the TCP flows within each fluid class are treated the same with the same congestion window behavior; this would amplify the changes in the aggregate flow rates, because the congestion windows would increase or decrease

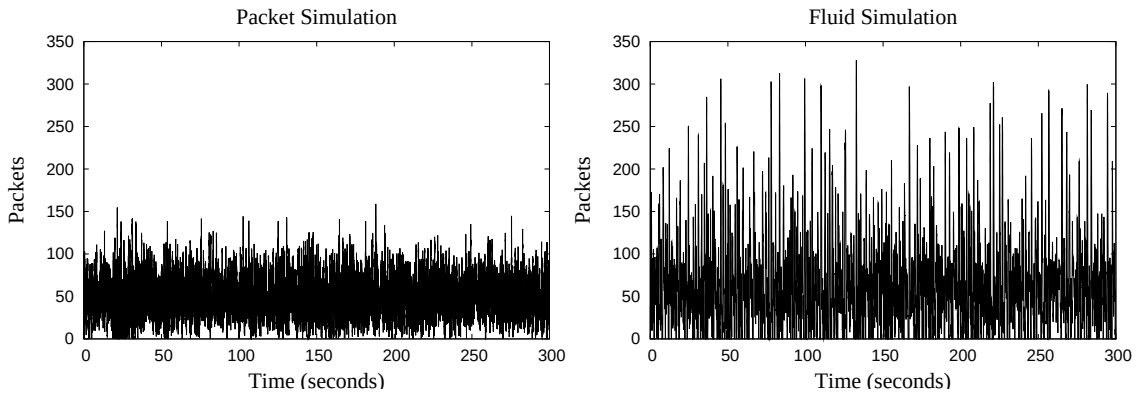


Figure 3.5: Bottleneck queue lengths from packet vs. fluid models

simultaneously among all flows within each fluid class. The synchronization effect is an artifact inherent to the fluid model. We are currently investigating ways to remove this problem.

3.3.3 Application Behavior

We validate our background fluid model baseline by studying the application behavior in the presence of the fluid background traffic and comparing the results against those from the packet simulation. We select two kinds of applications, file download and multimedia streaming, for this validation.

We use the same background traffic as described in the previous experiment. We add another server host on the left of the dumbbell and another client host on the right. For file download, we set the server to transfer a 30 MB file to the client and measure data received at the client side. For data streaming, we use a constant bit-rate UDP application and measure the delay jitter between consecutive packet arrivals. The time interval between the streaming data sent from the server is 8 ms, which corresponds to a 1 Mb/s send rate.

Figure 3.6 shows the cumulative fraction of the file download time. The average download time of fluid model is about 10% more than that of the packet model, which is probably due to the more pronounced variation in the queue length in fluid model.

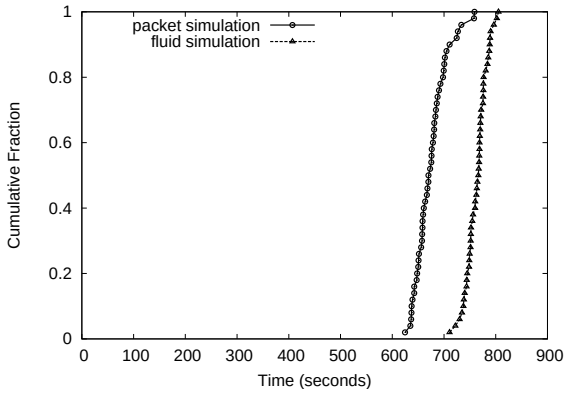


Figure 3.6: Download fraction.

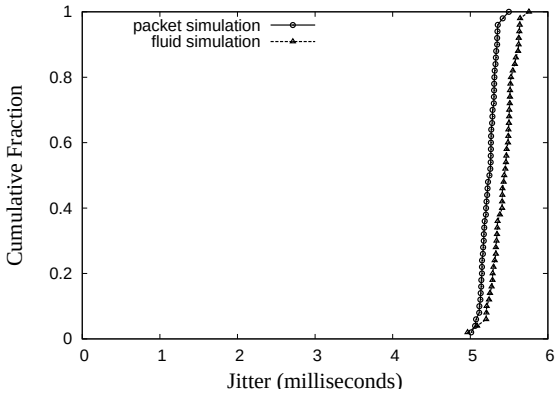


Figure 3.7: Delay jitter.

Figure 3.7 shows the cumulative fraction of the delay jitter of the arrived streaming data at the client side, which also reflects the higher variation in the fluid model.

3.4 Conclusion

Fluid models, more particularly, hybrid models that combine both fluid and packet representations, provide a cost-effective alternative to pure discrete-event packet simulation for rendering Internet traffic conditions needed for testing distributed applications, network protocols, and services. The performance gain comes from a certain level of abstraction. However, unrealistic abstractions may lead to incorrect evaluation results. The work in this chapter enhanced an existing hybrid model by removing two unrealistic assumptions existing in the fluid part. The improved background traffic model can correctly capture a good level of the important realism from the Internet observation of a single link as with the pure packet-oriented approach, while still maintaining a significant performance advantage.

CHAPTER 4

A FAST RATE-BASED TCP BACKGROUND TRAFFIC MODEL

Due to the complexity of network traffic, it is unlikely to have one all-purpose background traffic model. The best practice is to apply the proper background traffic model based on the goal of the specific network study. Therefore, there exist different background traffic aiming to model different aspects of the network traffic. This chapter presents a fast rate-based TCP (RTCP) traffic model designed to reduce the time and space complexity for simulating network traffic and maintain good accuracy.

4.1 Introduction

Since TCP handles the brunt of today's network traffic, a good aggregate traffic model must be able to accurately capture the TCP behavior. One type of solution (e.g., [MGT00] and [MGT03]) simulates TCP by only capturing the long-term average behavior of TCP in order to improve computational efficiency. This approach inevitably results in a loss of accuracy as it overlooks the fine-grained time varying features of TCP. The other type of solution (e.g., [Nic01] and [NY04]) maintains the detailed state transitions of TCP, in particular, the evolution of the TCP congestion window at the senders. However, in doing so it can greatly inflate the computational demand.

In this work, we present a fast rate-based model to approximate the TCP behaviors, which offers a new level of granularity for simulating TCP traffic. Our solution uses the packet aggregation idea, similar to those employed in [AD96], [GGT00], and [NY04]. In particular, rather than modeling the network traffic packet by packet as in the traditional approach, we group the packets as chunks of variable length, called *rate windows*, which consist of a number of packets from the same TCP session and with a similar arrival rate. We describe each rate window by the packet arrival rate and duration, both of which can be

adjusted according to the network condition as the chunk of packets visit the routers along the path from the source to the destination altogether. The model calculates the queuing delays and the packet losses as these rate windows traverse the individual network queues along the flow path.

A distinct feature of the proposed model is that the TCP congestion control behavior is represented using analytical models in response to the simulated network traffic conditions at different phases of a TCP connection. That is, rather than modeling the complex state transitions of TCP, we apply mathematical models to derive the packet send rate as a function of the round-trip time and the packet loss rate. The round-trip time and the packet loss rate are measured along the flow path as the rate windows are sent to visit the network queues and interact with other flows represented either as rate windows or individual packets. The latter, in particular, allows the model to seamlessly mix with a packet-oriented simulation. The proposed RTCP model is able to achieve a performance advantage over other TCP models, by integrating analytical solutions and aggregating traffic using rate windows. This comes at the cost of potential degradation of accuracy. Experiments show that the RTCP model is able to correctly capture the overall TCP behavior. Even for congested networks, our model can reduce the amount of simulation events and accelerate simulation by more than two orders of magnitude. For uncongested network conditions, the RTCP model can improve execution time even more drastically.

4.2 Related Work

Ahn and Danzig [AD96] proposed the idea of "packet trains". Each packet train consists of a number of closely spaced packets. The time complexity of this approach depends on the size of the packet train. The modeling granularity can be adjusted to go between a packet-level simulation, where the train size is limited to contain only a single packet, and a conversation-level simulation, where the entire flow is modeled as a single packet

train. It has been shown that the packet train model can achieve a significant reduction in the number of simulation events over the corresponding packet-oriented simulation—by as much as an order of magnitude if using a coarse granularity.

Guo et al. [GGT00] proposed a time-stepping hybrid simulation (TSHS) framework. The packets that arrive within a time step are put together as a single "chunk". Instead of handling individual packets, the routers process the whole chunk as it arrives. At the flow destination, packets within a chunk are individually acknowledged. Similar to the packet train approach, the time complexity of this approach can be adjusted, in this case, by choosing the length of the time step. Compared with the traditional packet-oriented simulation, the hybrid model has been shown to achieve moderate speedup when using sufficiently large time steps. It has been shown that TSHS can achieve over 3 times speedup with a time step of 2 ms [GGT00].

Although both the packet train and TSHS employ aggregation, many packet-level details remain in the models. For example, the packet train model still emits individual packets at the source and converts them to packet trains on the fly. TSHS carries the information of individual packets in the chunk so that the packets can be separately acknowledged at the destination. In contrast, fluid models (e.g., [KSCK96], [MGT00], [LFG$^+$01], [Nic01], [MGT03], and [NY04]) eliminate the packet-level information altogether and only describe the flow-level traffic intensity.

The fluid models face difficulty in modeling the packet-level details of the fluid flows. The time-driven fluid models (e.g., [MGT00]) can only capture the average behavior of long-term TCP flows (during the TCP congestion avoidance mode). The models therefore cannot accurately represent the transient behavior of the TCP flows. The event-driven fluid models (e.g., [NY04]) address this problem by including complex logic to handle detailed TCP transactions in response to packet losses during the TCP slow start and congestion avoidance phases. This method, however, increases the computational demand.

Instead of flow-level aggregation, the aggregation of traffic in the gateway at access network level is proposed in [CDXL08]. This model treats the access network as one single node. The traffic generator generates the amount of traffic for the entire access network and pushes the traffic to transmission controller. The transmission controller estimates the network condition according to the feedback of the ACKs and sends the packets to the destination through the network. This method achieves better performance than the traditional packet-oriented simulation through the aggregation of individual senders within one access network. However, the traffic transactions through the network are still represented packet by packet. Therefore, the improvement of its performance is limited.

Most aggregate models discussed above can be extended to hybrid models that aim at capturing the interactions between fluid background traffic and foreground packet-oriented flows (e.g., [RJF02], [KSWU03], [GLT04], and [Liu06]). Therefore, their focus is on the integration mechanism, not the statistical behavior of Internet traffic.

To efficiently model the TCP behavior, analytical models have been widely used. These analytical models (e.g., [MSMO97], [CSA00], [PFTK00], [MSZ02], [SKV04], and [AAB05]) provide an approximation of the TCP congestion control behavior. Some models (e.g., [MSMO97], [PFTK00], and [AAB05]) only capture the steady-state behavior of TCP for long-lived flows; others only consider the connection establishment and slow-start phase of TCP for short-lived flows (e.g., [MSZ02]); while others model TCP transfers for any given sizes of flows (e.g., [CSA00], and [SKV04]). To the best of our knowledge, these analytical models have not been used extensively in network simulation. Our rate-based TCP traffic model incorporates the analytical TCP models to reduce the computational complexity at the end nodes. In particular, to deal with the arbitrary sizes of TCP flows, we use the models proposed in [CSA00] to predict the traffic intensity for both short and long-lived TCP flows as a function of the round-trip time and the packet loss rate. For simplicity, we use the model proposed by [PFTK00] to approximate the

TCP send rate in steady state. This approximation has also been applied in Cardwell et al.'s model as a solution for capturing the TCP behavior in congestion avoidance stage. The models we adopt in our model are well validated by using simulations, controlled Internet measurements, and comparisons with live traces [KT04]. Our solution also adopts the general packet aggregation idea from the packet train model [AD96], TSHS [GGT00] and the event-driven fluid model [NY04].

4.3 An Overview of the Model

The essential aspect of TCP is its congestion control mechanism. Since TCP congestion control is performed at end nodes, we use an RTCP sender and an RTCP receiver to regulate the data transfer in simulation. It is important to note that our model is based on three assumptions. First, we assume that the time it takes to send all packets in a congestion window is smaller than the round-trip time. This assumption is normally true for today's network with high bandwidth and long latency. Second, we assume that packet losses within a round (defined by the round-trip time) are independent of the losses in other rounds, while packet losses in the same round can be correlated. This assumption holds for FIFO drop-tail queues [BVG96, Pax97a, YMKT99], but may not be true for RED queues [FJ93] or for paths where packet loss is largely due to link errors rather than congestion [CSA00]. Finally, the ACK losses are insignificant and therefore ignored after the initial handshake. This assumption is acceptable because the ACK packets are much smaller than the normal data packets, which makes them less likely to be dropped due to congestion. As expected, network paths are often more congested in the direction from data sender to receiver than in the reverse direction [TMW97].

It is also important to note that our study here is limited to TCP Reno, which implements fast retransmit and fast recovery. In addition, we assume that the receiver implements delayed ACK—an ACK is sent for the successful delivery of two consecutive data

segments. These limitations are simply because of the analytical models which we use for estimating the send rate for the TCP flows. Our model can be extended in principle by substituting the analytical models with those for other TCP variants, such as BIC and CUBIC.

We model three distinct phases of TCP. *Connection establishment* consists of the three-way handshake used by TCP to establish the connection between the sender and the receiver. *Slow start* is the exponential growth phase, during which TCP aggressively increases its congestion window by the same number of acknowledged segments. *Congestion avoidance* is the primary phase for congestion control, where TCP implements the additive increase and multiplicative decrease scheme. The congestion avoidance phase may also include periods of slow start, where TCP recovers from detected packet losses. However, we do not model slow start after retransmission timeouts because of the analytical models we adopt here. Additionally, retransmission timeouts do not happen very often under low loss rate conditions. We do not model TCP termination phase because it does not play an important role in determining the throughput or latency of a data transfer [CSA00].

We define four types of messages between the RTCP sender and the RTCP receiver:

- The `START` messages are used as probes, which accumulate the round-trip time and packet loss rate during the TCP connection establishment and slow start.
- The `DATA` messages carry the data over the TCP connection from the RTCP sender to the RTCP receiver. Each `DATA` message contains a "rate window", which represents a chunk of packets considered to have the same arrival rate. A `DATA` message keeps track of the packet arrival rate, the duration, as well as the delivery ratio for the rate window, which can be adjusted as the message travels over the network from the sender to the receiver.

- The UPDATE messages are sent from the RTCP receiver or an intermediate router to inform the RTCP sender of the network condition so that the sender can adjust the TCP congestion window and send rate accordingly.
- An END message is sent from the RTCP receiver to the RTCP sender once the receiver has successfully received all the data. The END message will cause the sender to immediately stop the transmission.

To guarantee that the RTCP sender is aware of the network conditions, we do not allow START, UPDATE, nor END messages to be dropped in simulation. Both START and DATA messages are treated as regular packets, which means they experience proper queueing delays in the network. UPDATE and END messages, however, are special packets which are delivered to the RTCP sender directly and are never queued.

To reduce the computational cost of the simulation, we model TCP data transfer in the unit of "rate windows", where a number of consecutive packets from the same TCP session are lumped together and described using a constant arrival rate. The changes in rate of each rate window are determined by the network conditions (i.e., RTT and loss rate) that the previous rate window of the same flow has experienced. The RTT and loss rate are both rate-window specific. When loss happens, we do not keep track of which packet (or packets) are lost within a given rate window, instead we update the delivery ratio to indicate the loss. Each DATA message carries one rate window sent from the RTCP sender to the RTCP receiver. A rate window has the following attributes:

- *Send time*: the time at which the rate window is sent out from the RTCP sender. The send time will be copied to the UPDATE message on the way back to the RTCP sender, which uses the UPDATE message to calculate the round-trip time.

- *Arrival rate*: the rate at which packets belonging to the rate window are sent. At the beginning, the arrival rate is the same as the send rate of the TCP sender. In the subsequent queues, the arrival rate is the departure rate at the preceding queue.
- *Duration*: the length of the rate window in time. The product of the arrival rate and the duration is the total number of packets represented by the rate window, which remains unchanged as the rate window travels from the sender to the receiver.
- *Delivery ratio*: the proportion of data successfully delivered by the rate window so far. It is the number of bytes remaining in a rate window over the total number of bytes originally sent from the RTCP sender. At the beginning, the delivery ratio is one; and the delivery ratio may decrease as a result of packet losses at the intermediate routers.

Figure 4.1 shows the high-level interactions between the sender and the receiver. A TCP session starts when the RTCP sender sends a `START` message over the network to the RTCP receiver, which immediately sends it back (steps 1 and 2). This allows the sender to probe the network and calculate the expected duration of the handshake phase before a TCP connection can be successfully established. After that, the RTCP sender starts sending `DATA` messages, each representing one rate window with the packet arrival rate calculated from measured round-trip time and the packet loss probability. For each `DATA` message received, the RTCP receiver returns an `UPDATE` message carrying the necessary information for the RTCP sender to get the round-trip time and the packet loss probability (steps 3 and 4).

Each intermediate network queue on the flow path from the sender to the receiver can modify the arrival rate, the duration, and delivery ratio of the `DATA` message, as the rate window interacts with other rate windows. If somehow the delivery ratio of a `DATA` message drops to zero, the entire rate window is considered lost, in which case an

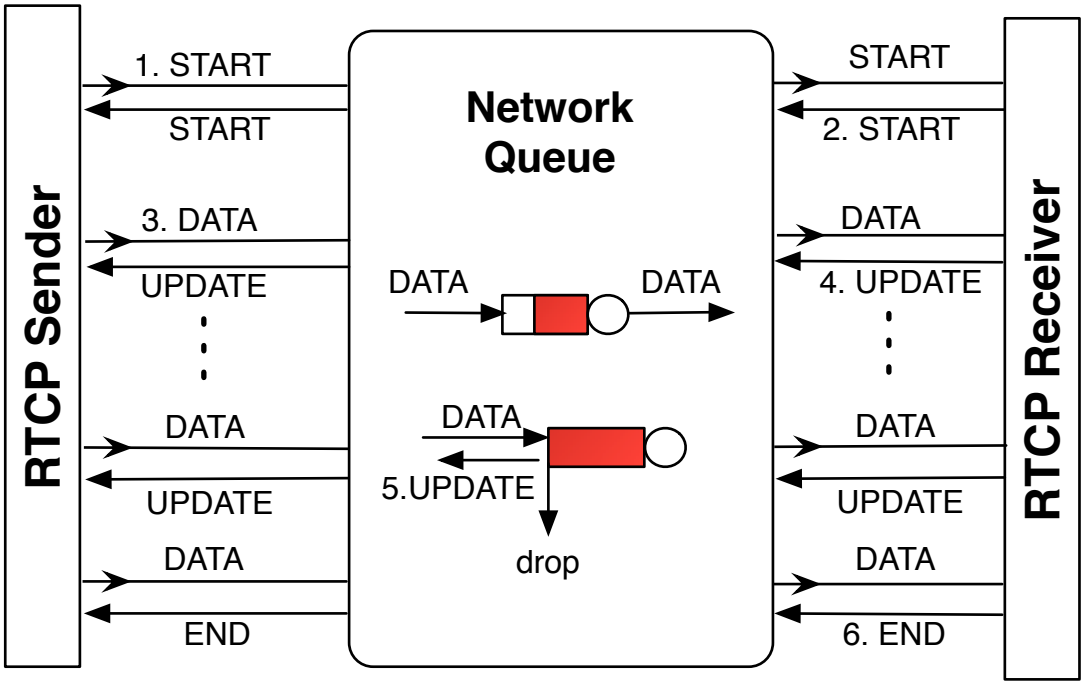


Figure 4.1: Interactions between RTCP sender and receiver

`UPDATE` message is immediately sent back to the RTCP sender so that the sender can be notified of the loss and adjust its send rate quickly (step 5). The RTCP receiver is responsible for determining whether a data transfer has completed (the data transfer size has been given to the receiver during the connection establishment), and if so, the RTCP receiver sends an `END` message to the sender to stop further transmissions (step 6).

The cost of the RTCP model depends on the size of the rate window. If the rate window is too small, we cannot achieve a significant performance improvement over the traditional packet-oriented simulation. If the rate window is too large, RTCP may not be able to properly respond to network congestions and cause significant errors in its estimation of the throughput. We choose to use the round-trip time as the duration of the rate window, which means that the size of a rate window is equal to the current congestion window size. This is a reasonable choice because TCP cannot react to network

congestions unless the packet is delivered and acknowledged (or timeout happens). As an optimization, if we have prior knowledge that the network is under congested, that is, if the network has sufficient resources so that traffic experiences little loss, we can increase the size of the rate window for improved efficiency.

4.4 Determining Send Rate

We extend existing analytical models to represent the TCP congestion control behavior. In particular, during the connection establishment and slow start phases, we apply the model proposed by [CSA00] to estimate the duration of the phases and the send rate. We use the model proposed by [PFTK00], which is also a part of Cardwell et al.'s model [CSA00], to estimate the send rate during the congestion avoidance phase. Both models describe the TCP congestion window behavior as a function of the round-trip time and the packet loss rate. Both TCP timeout and triple duplicate ACKs are taken into account in these two models. In the following, we elaborate on the analytical models during the three TCP phases.

4.4.1 Connection Establishment

Connection establishment is the first phase of TCP, which consists of the three-way handshake before a successful TCP connection can be established. The three-way handshake consists of zero or more failed attempts by the sender to transmit the TCP SYN message, followed by one successful delivery of the TCP SYN message, and then zero or more failed attempts to transmit the TCP SYN/ACK message by the receiver, followed by one successful delivery of the SYN/ACK message. The expected number of failed attempts depends on $p_f(t)$ and $p_r(t)$, which are defined to be the packet loss probability on the forwarding path (from the sender to the receiver) and on the reverse path (from the receiver

to the sender) at time t, respectively. Typically, TCP will give up connection attempts after 4 to 6 failures. If $p_f(t)$ and $p_r(t)$ are low, most TCP handshakes will be successful before giving up. Using the analytical model proposed in [CSA00], we can estimate the duration of a successful connection establishment, $T_{ce}(t)$, as follows:

$$T_{ce}(t) = \pi(t) + t_s \left(\frac{1 - p_r(t)}{1 - 2p_r(t)} + \frac{1 - p_f(t)}{1 - 2p_f(t)} - 2 \right) \tag{4.1}$$

where $\pi(t)$ is the measured round-trip time, and t_s is the length of the TCP timeout for the SYN message.

We set t_s to be 3 seconds, which is a typical value [Bra89]. The round-trip time, $\pi(t)$, and the packet loss probabilities, $p_f(t)$ and $p_r(t)$, are collected during the exchange of the `START` messages between the RTCP sender and receiver. Once the expected duration of the connection establishment is determined, the RTCP sender waits for the duration before it enters the slow start phase, which we describe next.

4.4.2 Slow Start

In the slow start mode, TCP quickly increases its congestion window until it detects a packet loss. From now on, we assume that packet loss only happens in the forwarding path from the sender to the receiver. Suppose the flow size is ξ. If there is no packet loss, i.e., $p_f(t) = 0$, we expect to send all ξ segments during the slow start phase. However, if $p_f(t) > 0$, according to [CSA00], the expected number of segments to be sent during the slow start phase, $L_{ss}(t)$, can be estimated as follows:

$$\begin{aligned} L_{ss}(t) &= \sum_{k=0}^{\xi-1} (1 - p_f(t))^k p_f(t) k + (1 - p_f(t))^{\xi} \xi \\ &= (1 - (1 - p_f(t))^{\xi})(1 - p_f(t))/p_f(t) + 1 \end{aligned} \tag{4.2}$$

Here, we assume that the size of a segment (i.e., a TCP packet) is the maximum segment size (MSS).

If the TCP congestion window is not constrained by the maximum window size W_{max}, the result TCP congestion window after sending L_{ss} segments can be approximated by:

$$W_{ss}(t) = L_{ss}(t)(\gamma - 1)/\gamma + W_1/\gamma \tag{4.3}$$

where W_1 is the initial value of the sender's congestion window, and γ denotes the rate of exponential growth of the congestion window during the slow start. Typically, $1 \leq W_1 \leq 3$. We set γ to be 1.5 to capture the expected TCP behavior at slow start: the TCP sender increases its congestion window size by one MSS for each ACK packet received; and the TCP receiver that implements delayed ACK sends one ACK roughly for every two data packets. That is, during each round-trip time, the congestion window size increases by as much as half of the congestion window size during the previous round. Therefore, $\gamma = 1.5$.

To determine whether the TCP congestion window is constrained by the maximum send window size, we compare $W_{ss}(t)$ and W_{max}. If $W_{ss}(t) > W_{max}$, TCP experiences two phases during the slow start: at first, the congestion window size increases from W_1 to W_{max} at the rate γ per round-trip time; after that, the congestion window remains at W_{max}. If $W_{ss}(t) \leq W_{max}$, TCP would send all $L_{ss}(t)$ segments in the first phase.

The model proposed in [CSA00] predicts the duration of the slow start phase, $T_{ss}(t)$, as follows:

$$T_{ss}(t) = \begin{cases} \pi(t)(\log_\gamma(\frac{W_{max}}{W_1} + 1) + \\ \quad \frac{1}{W_{max}}(L_{ss}(t) - \frac{\gamma W_{max} - W_1}{\gamma - 1})) & \text{if } W_{ss}(t) > W_{max} \\ \pi(t)\log_\gamma(\frac{L_{ss}(t) * (\gamma - 1)}{W_1} + 1) & \text{otherwise} \end{cases} \tag{4.4}$$

We model the slow start phase by having the RTCP sender and receiver first exchange the `START` messages, from which we can use Equation (4.2) to determine $L_{ss}(t)$, the expected number of segments to be sent during the slow start phase. Then we use Equation (4.3) to determine $W_{ss}(t)$, the unconstrained congestion window size, and then use

Equation (4.4) to determine $T_{ss}(t)$, the expected duration of the slow start phase. If it is determined that the TCP session is still in the slow start phase, i.e., if the number of segments sent by TCP is less than $L_{ss}(t)$, for each round-trip time $\pi(t)$, the RTCP sender sends a `DATA` message carrying a rate window with the packet arrival rate $R_{ss}(t) = (L_{ss}(t)/T_{ss}(t))$. Whenever the RTCP sender receives an `UPDATE` message, it recalibrates $L_{ss}(t)$, $T_{ss}(t)$ and $R_{ss}(t)$. When the number of sent segments has gone beyond the expected value, TCP switches to congestion avoidance.

Our model may not be able to capture the exact time of the transition from slow start to congestion avoidance. This is because the model handles packet transfers in the unit of rate windows. The model does not keep track of the exact packet positions within the rate windows, and therefore it is impossible to record the exact time the congestion avoidance model would start. The error is considered insignificant since it depends on the size of the rate window, which is the round-trip time.

4.4.3 Congestion Avoidance

TCP enters steady state during the congestion avoidance phase. We use the model proposed in [PFTK00] to estimate the send rate as a function of the round-trip time, the packet loss (whether it is due to duplicate ACKs or timeouts), and the current congestion window size. According to the model, the send rate can be determined as follows:

$$R_{ca}(t) = \min\{\frac{W_m(t)}{\pi(t)}, R(t)\} \tag{4.5}$$

$R(t)$ is a rate determined by the round-trip time $\pi(t)$, and the packet loss probability on the forwarding path $p_f(t)$ as follows:

$$R(t) = \frac{1}{\pi(t)\sqrt{\frac{2bp_f(t)}{3}} + t_O \min\{1, 3\sqrt{\frac{3bp_f(t)}{8}})p_f(t)(1+32p_f^2(t))\}} \tag{4.6}$$

where t_O is the length of the timeout and b is the average number of packets that are acknowledged by each received ACK. We set $t_O = 0.2$ s and $b = 2$.

$W_m(t)$ is the current congestion window size. The model originally proposed in [PFTK00] does not provide a way to calculate the current congestion window. We extend the model by incorporating TCP's additive increase and multiplicative decrease behavior. We calculate W_m in rounds. At first we set W_m^0 to be the maximum congestion window size W_{max}. At round *i*, we update the congestion window W_m^i as follows:

$$W_m^i = \begin{cases} \min\{W_{max}, W_m^{i-1} + 1\} & \text{if } p_f(t) = 0 \\ \max\{1, W_m^{i-1}/2\} & \text{otherwise} \end{cases} \tag{4.7}$$

The TCP send rate is controlled by a closed loop system. A change in the measured round-trip time and packet loss rate will cause the RTCP sender to adjust its send rate. The RTCP receiver is responsible for sending the information back to the sender. During the connection establishment phase, the receiver immediately returns a `START` message after it receives a `START` message from the sender. During the data transfer (at the slow start and congestion avoidance phases), the receiver sends an `UPDATE` message for each received `DATA` message to the sender to report the current round-trip time and the packet loss rate. When the receiver determines that it has received all the data, the receiver sends an END message to inform the sender to terminate further transmission.

4.5 Conducting Flows at Queues

Being able to capture the interaction among the traffic flows at the network routers both accurately and efficiently is essential to the success of the RTCP model. In this section we show how the network queues react to rate windows. Our model is restricted to FCFS queues with drop-tail behavior. We first consider a single flow at the queue and then extend the model to handle the interaction among multiple flows.

4.5.1 Single Flow

We start by considering a drop-tail queue being fed by one and only one flow. Suppose a rate window arrives at an empty queue with an arrival rate λ^{in}, a duration δ^{in}, and a delivery ratio τ^{in}. In this case, $\lambda^{in}\delta^{in}$ is the total amount of data originally sent by the RTCP sender for the rate window (it is an invariant); $\lambda^{in}\delta^{in}\tau^{in}$ is the total amount of data actually arriving at the the queue during the rate window; and $\lambda^{in}\tau^{in}$ is the effective arrival rate. Let μ be the link capacity, which is the service rate of the queue.

If the effective arrival rate $\lambda^{in}\tau^{in}$ is less than the service rate μ, there will be no accumulation at the queue, and the attributes of the rate window will not change: $\lambda^{out} = \lambda^{in}$, $\delta^{out} = \delta^{in}$, and $\tau^{out} = \tau^{in}$.

Otherwise, if $\lambda^{in}\tau^{in} \geq \mu$, the rate window will be served at the maximum capacity μ and the queue will start to accumulate a back log at a rate equal to the difference between the effective arrival rate, $\lambda^{in}\tau^{in}$, and the service rate, μ. Suppose the buffer size is B. The final queue length when the rate window completes its arrival at the queue can be calculated as:

$$\hat{q}_0 = \left[(\lambda^{in}\tau^{in} - \mu)\delta^{in}\right]_0^B \tag{4.8}$$

where $[x]_0^B = \min\{\max\{x,0\},B\}$. The duration of the output rate window needs to be extended to include the time to flush out the back log (at the rate of μ):

$$\delta^{out} = \delta^{in} + \hat{q}_0/\mu \tag{4.9}$$

Since the product of the arrival rate and the duration should indicate the total amount of data for the rate window sent from the RTCP sender, the arrival rate of the output rate window can be easily adjusted:

$$\lambda^{out} = \frac{\lambda^{in}\delta^{in}}{\delta^{out}} \tag{4.10}$$

The delivery ratio needs to be calculated to account for the losses due to possible buffer overflow. In case that $(\lambda^{in}\tau^{in} - \mu)\delta^{in} \leq B$, the rate window will not experience any

loss and therefore $\tau^{\mathsf{out}} = \tau^{\mathsf{in}}$. Otherwise, there will be data loss due to buffer overflow:

$$\hat{\theta} = ((\lambda^{\mathsf{in}}\tau^{\mathsf{in}} - \mu)\delta^{\mathsf{in}} - \mathsf{B}) \tag{4.11}$$

The delivery ratio can be updated as follows:

$$\begin{aligned} \tau^{\mathsf{out}} &= \frac{\lambda^{\mathsf{in}}\tau^{\mathsf{in}}\delta^{\mathsf{in}} - \hat{\theta}}{\lambda^{\mathsf{in}}\delta^{\mathsf{in}}} && (4.12) \\ &= \frac{\lambda^{\mathsf{in}}\tau^{\mathsf{in}}\delta^{\mathsf{in}} - [(\lambda^{\mathsf{in}}\tau^{\mathsf{in}} - \mu)\delta^{\mathsf{in}} - \mathsf{B}]}{\lambda^{\mathsf{in}}\delta^{\mathsf{in}}} \\ &= \frac{\mathsf{B} + \mu\delta^{\mathsf{in}}}{\lambda^{\mathsf{in}}\delta^{\mathsf{in}}} && (4.13) \end{aligned}$$

4.5.2 Multiple Flows

Now we consider the more interesting case when the queue consists of multiple flows. We define $\mathsf{W}_\mathsf{i}(\lambda_\mathsf{i}, \delta_\mathsf{i}, \tau_\mathsf{i}, \mathsf{s}_\mathsf{i}, \mathsf{e}_\mathsf{i})$ to be a rate window currently visiting the network queue, where λ_i is the packet arrival rate, δ_i is the duration, τ_i is the delivery ratio, s_i is the start time, and e_i is the end time ($\mathsf{e}_\mathsf{i} = \mathsf{s}_\mathsf{i} + \delta_\mathsf{i}$). In the algorithm, we maintain a priority queue Q to store all current rate windows visiting the network queue; we use the end time e_i as the priority key. Initially, the priority queue is empty. We use A to store the aggregate effective arrival rate of all rate windows, i.e., $\mathsf{A} = \sum_{\mathsf{i}\in\mathsf{Q}} \lambda_\mathsf{i}\tau_\mathsf{i}$. A is initialized to be zero. We denote t_0 to be the arrival time of the previous rate window, and q_0 to be the queue size at that time. Both t_0 and q_0 are initialized to zero. We assume the capacity of the network queue is B, and the delay of the link between the network queue and the subsequent queue is D.

Alg. 1 (on page 58) shows the algorithm that processes the simulation event representing the arrival of a new rate window W_f at time s_f (which is the current simulation time). An example is given in Fig. 4.2, where rate window f arrives at the queue at time s_f. In this example, the previous rate window arrived at the queue is rate window 3. Assume that the algorithm correctly sets $\mathsf{t}_0 = \mathsf{s}_3$, $\mathsf{q}_0 = \mathsf{q}(\mathsf{s}_3)$, and $\mathsf{A} = \sum_{\mathsf{k}=1}^{3} \lambda_\mathsf{k}\tau_\mathsf{k}$, after the previous

Algorithm 1 Process arrival of rate window $W_f(\lambda_f, \delta_f, \tau_f, s_f, e_f)$

1: **while** (Q is not empty) **do**
2: $W_i(\lambda_i, \delta_i, \tau_i, s_i, e_i) \leftarrow$ Q.peekMin()
3: **if** $(e_i > s_f)$ **then break**
4: $q_0 \leftarrow [q_0 + (A - \mu)(e_i - t_0)]_0^B$; $t_0 \leftarrow e_i$
5: $A \leftarrow A - \lambda_i \tau_i$
6: Q.deleteMin()
7: $q_0 \leftarrow [q_0 + (A - \mu)(s_f - t_0)]_0^B$; $t_0 \leftarrow s_f$
8: $A \leftarrow A + \lambda_f \tau_f$
9: $\hat{q}_0 \leftarrow q_0$; $\hat{t}_0 \leftarrow t_0$; $\hat{A} \leftarrow A$
10: $S \leftarrow \phi$; $\hat{\theta} \leftarrow 0$
11: **while** (Q is not empty) **do**
12: $W_i(\lambda_i, \delta_i, \tau_i, s_i, e_i) \leftarrow$ Q.peekMin()
13: **if** $(e_i > e_f)$ **then break**
14: $\hat{q}_0 \leftarrow [\hat{q}_0 + (\hat{A} - \mu)(e_i - \hat{t}_0)]_0^\infty$; $\hat{t}_0 \leftarrow e_i$
15: **if** $(\hat{q}_0 > B)$ **then**
16: $\hat{\theta} \leftarrow \hat{\theta} + (\hat{q}_0 - B)\lambda_f \tau_f / \hat{A}$
17: $\hat{q}_0 \leftarrow B$
18: $\hat{A} \leftarrow \hat{A} - \lambda_i \tau_i$
19: Q.deleteMin()
20: S.insert($W_i(\lambda_i, \delta_i, \tau_i, s_i, e_i)$)
21: $\hat{q}_0 \leftarrow [\hat{q}_0 + (\hat{A} - \mu)(e_f - \hat{t}_0)]_0^\infty$
22: **if** $(\hat{q}_0 > B)$ **then**
23: $\hat{\theta} \leftarrow \hat{\theta} + (\hat{q}_0 - B)\lambda_f \tau_f / \hat{A}$
24: $\hat{q}_0 \leftarrow B$
25: **for all** $w \in S$ **do** Q.insert(w)
26: Q.insert($W_f(\lambda_f, \delta_f, \tau_f, s_f, e_f)$)
27: $\delta_j \leftarrow \delta_f + \hat{q}_0/\mu$; $\lambda_j \leftarrow \lambda_f \delta_f / \delta_j$; $\tau_j \leftarrow (\lambda_f \tau_f \delta_f - \hat{\theta})/(\lambda_f \delta_f)$
28: $s_j \leftarrow s_f + q_0/\mu + D$; $e_j \leftarrow s_j + \delta_j$
29: schedule arrival of $W_j(\lambda_j, \delta_j, \tau_j, s_j, e_j)$ at next queue

invocation of the algorithm. The priority queue currently should contain the three rate windows (sorted by their end times).

The algorithm starts by calculating the current queue length (lines 1-9). If there are rate windows in the priority queue Q that finish before the current time s_f, the algorithm adjusts the queue length q_0 (line 4) and the aggregate effective arrival rate A (line 5), to simulate the effect of these rate windows in the increasing order of the end time. The rate windows are also removed from the priority queue Q (line 6), since they will be no longer needed. After that, the algorithm updates the current queue length (line 8) and adds the

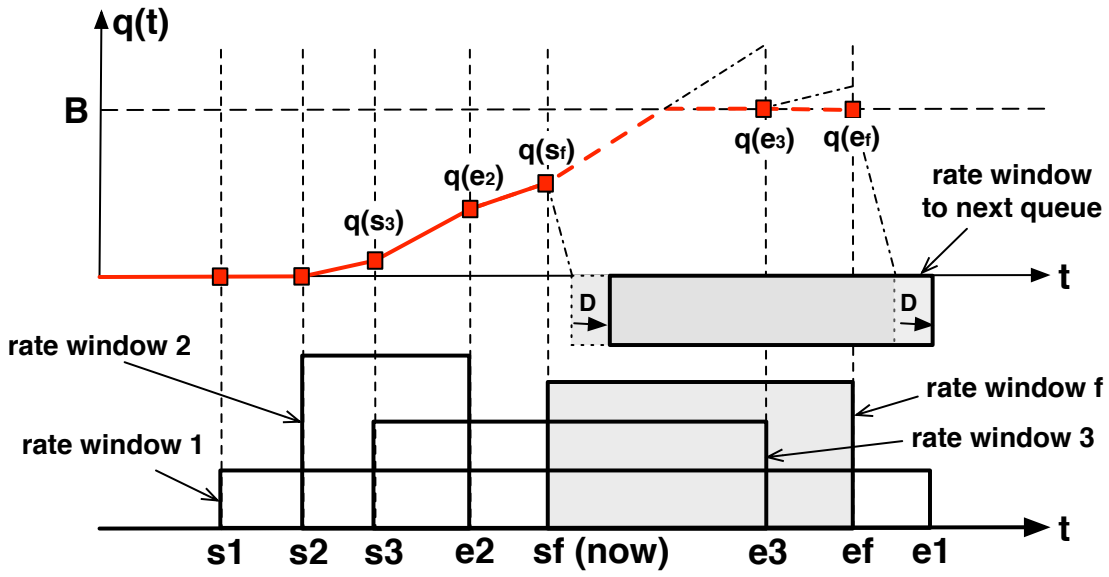


Figure 4.2: An example showing the queuing length changes at the start and end of rate windows

effective arrival rate of the new rate window to the aggregate rate (line 9). In the example, rate window 2 is only one that completes after s_3 and before s_f. Therefore, we update q_0 from $q(s_3)$ to $q(e_2)$ and then from $q(e_2)$ to $q(s_f)$; the example assumes that the aggregate effective arrival rate at the two segments is larger than the service rate; that's why the queue length increases.

Next, the algorithm needs to compute the *projected* queue length at the end time of the newly arrived rate window. The queue length $\hat{q}_0$ is used to calculate the duration of the output rate window (see Equation 4.9). The algorithm also needs to compute the *projected* data loss due to buffer overflow. The data loss $\hat{\theta}$ is used to update the delivery ratio of the output rate window (see Equation 4.12).

To compute the projected values, the algorithm needs to scan into the simulated future. In order to protect the queue length, the current time, and the aggregate effective arrival rate from being overwritten, the algorithm first creates a set of shadow variables

and copies the values from the original variables (line 10). The algorithm also uses a set S to temporarily store the rate windows removed from the priority queue; these rate windows need to be restored once the scan is finished (lines 22 and 29). The algorithm uses $\hat{\theta}$, which is initialized at line 11, to accumulate the amount of data loss due to buffer overflow.

If there are rate windows in the priority queue Q that finish before the end time of the newly arrived rate window, e_f, the algorithm deals with them one by one in increasing end time order. The algorithm first adjusts the shadow queue length $\hat{q}_0$ assuming the queue has infinite capacity (line 15). If the shadow queue length turns out to be larger than the buffer size (lines 16-19), the surplus is deemed to be lost due to overflow. The quantity is then proportioned according to the effective arrival rate and added to accumulative data loss (line 17). The shadow queue length is then reset to the maximum buffer size (line 18). The algorithm also updates the shadow aggregate effective arrival rate $\hat{A}$ as the rate window ends (line 20). After that, the algorithm updates the shadow queue length at the end of the current rate window (line 24). The same logic is applied to calculate the data loss (lines 25-28).

In the example, rate window 3 ends before the newly arrived rate window ends. Therefore, the shadow queue length $\hat{q}_0$ is updated from $q(s_f)$ to $q(e_3)$ and then to $q(e_f)$. Buffer overflow happens both at e_3 and e_f; the surplus is added proportionally to the accumulative data loss $\hat{\theta}$.

The algorithm finally inserts the newly arrived rate window into the priority queue (line 30). In the mean time, it creates the output rate window with the updated attributes (line 31); the output rate window will be sent downstream to the next hop. The start time of the output rate window is calculated to be the start time of the input rate window s_f, plus the time needed to flush all data enqueued at time s_f (right before the input rate window arrives), and the propagation delay between this queue and the one downstream (line 32).

The algorithm schedules an event to represent the arrival of the output rate window at the next queue (line 33).

We note that the algorithm calculates the projected queue length and data losses based only on the interaction between the newly arrived rate window and the existing rate windows in the queue. In particular, it assumes that future arrivals will not alter the prediction. This is obviously an approximation. If a rate window later arrives at a congested queue (resulting an aggregate arrival rate larger than the service rate), every flow in the queue needs to adjust its output rate for a fair share of the bandwidth. That is, one rate change may result in the update of all the rate windows at this queue as well as all subsequent queues. This phenomenon is called the "ripple effect" and has been well documented [Nic01, LFG+01].

The ripple effect can significantly increase the computational demand. To avoid that, in our model, we do not allow the rate windows to be changed once they have been propagated to the downstream queues. We only make changes to the newly arrived rate windows and project the queuing effect based on the rate windows currently in the queue. Our solution is reasonable because our model is a close-loop system: the queuing state is updated continuously as the RTCP sender adjusts its send rate at each round and sends a `DATA` message carrying a rate window and subsequently receives the `UPDATE` message carrying the network measurements. Even if the current rate window in a queue does not immediately adjust its rate to respond to the succeeding arrivals that overlap with it, the rate window in the next round will.

We need to be aware that such approximation may result in a rare condition where the rate window sent in the next round (from the same sender to the same receiver) catches up with the rate window in the previous round at a queue. This condition is caused by the over-estimation of the round-trip time in the previous round. Since the changes to the existing rate window is not propagated, it is possible that the sender sends the rate window

for the next round with a start time earlier than the end time of the previous rate window. The successive rate windows of the same flow may overlap. To compensate this error, we simply terminate the previous rate window in the queue as it carries stale information.

4.6 Experiments

We implemented the RTCP model in PRIME [PRI11], which is a parallel simulator designed for simulating large-scale network models. PRIME provides detailed models of 14 TCP congestion control algorithms, including RENO, BIC, and CUBIC, which have been validated carefully through extensive studies [ELL09]. PRIME also supports real-time simulation, where simulated network protocols can interact with real network devices. The following experiments were conducted on a Linux workstation with a 2.3 GHz Intel Core2 Duo processor and 2 GB of RAM. All measurements shown are averages of 20 trials.

4.6.1 Dumbbell Topology

Our first set of experiments aim to provide a baseline comparison between RTCP and the detailed TCP models. We use a simple dumbbell network (shown in Fig. 4.3), which consists of 2 routers and 4 hosts with 2 servers and 2 clients. The connection between the 2 routers forms a bottleneck link, configured with 10 Mb/s bandwidth and 64 *ms* delay. We set the bandwidths of all other links to be 1 Gbps, and set their delays to be 1 μs. All network interfaces use drop-tail queues with a buffer size of 70 KB (around 50 packets). Using this dumbbell model, we study RTCP's accuracy and performance in terms of speedup and reduction in event count.

First, we consider the scenario with a single flow. At the start of the simulation, we direct one TCP flow from S_1 to C_1. We observe the queue length and loss at R_1; we also measure the overall throughput of the TCP flow. The results, shown in Table 4.1,

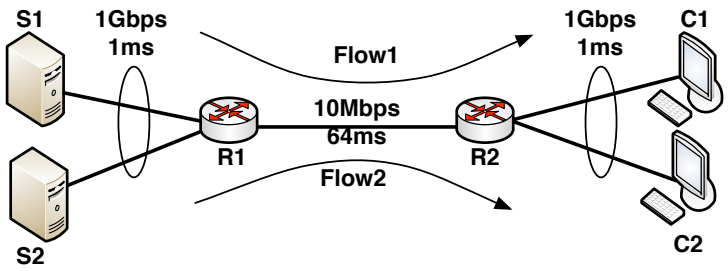


Figure 4.3: Dumbbell network with two flows

Table 4.1: Statistics for Dumbbell Network with One Flow

	TCP	RTCP
Throughput (Mb/s)	8.938	9.111
Average Loss Probability	0.08%	0.03%
Average Queue Length (packet)	15.6	8.2
Simulation Events	361,065	5,556
Exec Time Avg $\pm$ Std (sec)	1.404 ± 0.015	0.0152 ± 0.0001
Event Ratio	1	0.015
Execution Ratio	1	0.011

indicate that RTCP can accurately capture the overall throughput of a detailed TCP flow. However, RTCP's average loss probability and queue length differ from the detailed TCP model, although RTCP can capture the overall queue behavior, as is seen in Fig. 4.4a.

We next study the scenario with 2 flows. At the start of the simulation, we direct one TCP flow from S_1 to C_1 and a competing flow from S_2 to C_2. The results are shown in Table 4.2. RTCP is able to capture the overall throughput and fairly divides the bandwidth between the two flows. RTCP is also able to capture the overall queue behavior, which is shown in Fig. 4.4b.

RTCP achieves a significant speed up in both scenarios. The number of events are reduced by a factor of 67 for a single flow and a factor of 43 for two flows. Likewise, the overall execution time is reduced by a factor of 91 and a factor of 53 for the single flow

Table 4.2: Statistics for Dumbbell Network with Two Flows

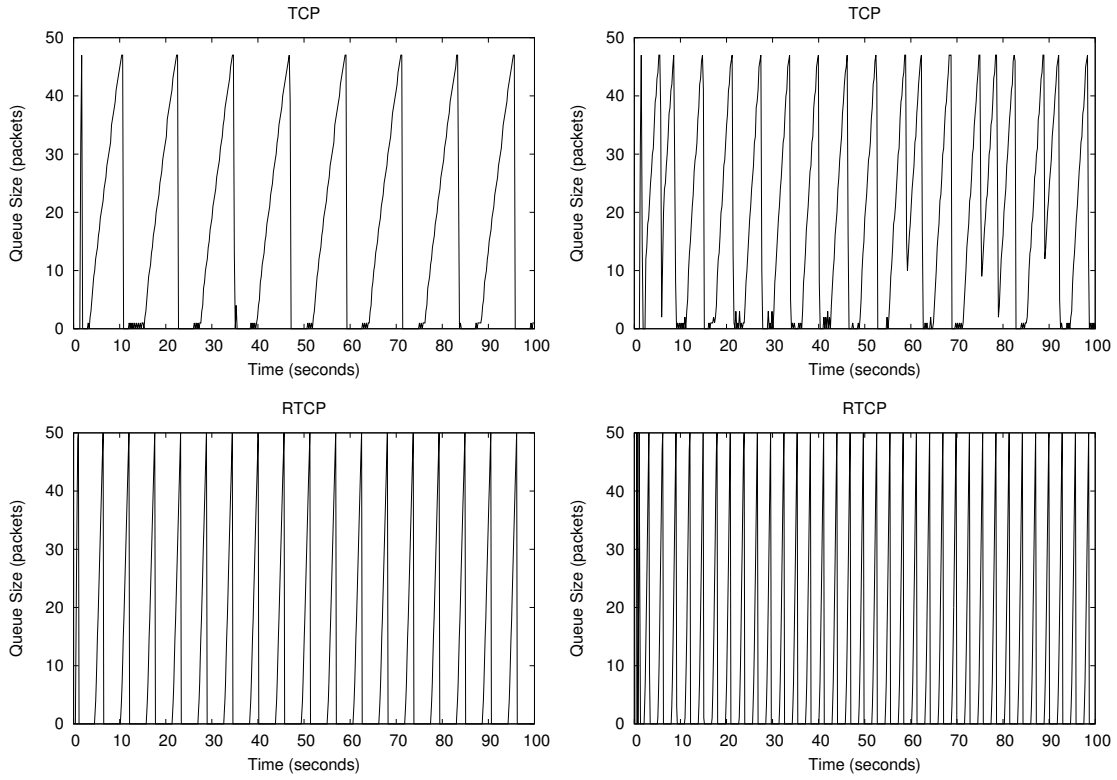
	TCP	RTCP
Flow 1's Throughput (Mb/s)	4.607	4.660
Flow 2's Throughput (Mb/s)	4.501	4.533
Aggregate Throughput (Mb/s)	9.107	9.193
Ave Loss Probability	0.16%	0.11%
Average Queue Length (packet)	18.7	9.5
Simulation Events	368,970	8,531
Exec Time Avg $\pm$ Std (sec)	1.453 ± 0.019	0.028 ± 0.0002
Event Ratio	1	0.023
Execution Ratio	1	0.019

and two flow cases. The two flow scenario sees a smaller speed up because RTCP must send more rate windows in response to the competition between the two flows.

4.6.2 Multiple Clients

Our next experiment examines the behavior of RTCP in the case where multiple clients download from a single server. The network model is depicted in Fig. 4.5. Each client (C_x) is connected with a link with delay set to be $1 + x * N$. N is the delay increment, which we vary in the experiment between 0 ms and 5 ms. For example, if we set $N = 2$, the adjacent links connecting clients and the router R will differ by a delay of 5 ms. The buffer size for all interfaces is configured to be 50 packets. The bottleneck link between the server S and router R has a bandwidth of 45 Mb/s and a delay of 64 ms. At the start of the simulation, each client initiates a download of a large file from the server.

Due to space limitation, we only show the results of the scenario when the delay increment is 0s in Table 4.3. To compare the model performance under all different scenarios as we change the delay increment, we show the results in Fig. 4.6 and Fig. 4.7. Fig. 4.6 shows the throughput achieved by each client using both the TCP and RTCP models after 100 seconds. Overall, RTCP and TCP match well. RTCP seems to achieve more balanced



(a) One flow (b) Two flows

Figure 4.4: Comparison of the instantaneous queue size at R_1

throughput than TCP (meaning it is more fair). As in the previous experiment, we also observed the average queue size at S for RTCP is consistently lower than that for TCP.

Fig. 4.7 shows the reduction in the number of events and in the execution time for the different values of N. On average, RTCP reduces the number of events by a factor of 18 and reduces the execution time by a factor of 70.

4.6.3 Multiple Bottleneck Model

In this experiment, we evaluate the performance of RTCP using the so-called "parking lot" model, which contains multiple bottleneck links. The topology, shown in Fig. 4.8, consists of 4 routers, 4 servers, and 4 clients. The bandwidths of all links are set to be 100 Mb/s. The delays of the links connecting routers and hosts are set to be 1 ms, and

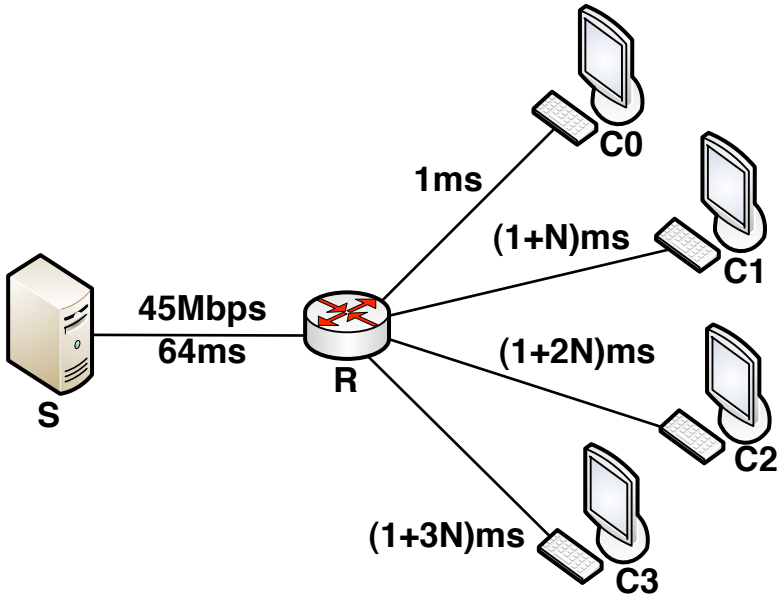


Figure 4.5: Multiple client network

the delays of the links between the routers are set to be 5 ms. Downloads are initiated by the clients with a 5 second offset. That is, Flow 0 starts at time 0, Flow 1 at 5s, Flow 2 at 10 s, and Flow 3 at 15 s. We initialize the flows at different times so that we can avoid the synchronization among the flows. The results are shown in Table 4.4. RTCP closely matches TCP's behavior in terms of the average and aggregate throughput. In this case, RTCP is able to reduce the number of events by a factor of 143 and the execution time by a factor of 125.

4.6.4 Large Scale Topology

For our last experiment, we evaluate RTCP in a large network scenario. To achieve the necessary scale and realism for our experiments, here we use BRITE for the backbone network and uses a campus network model for the network topology at the institutional level. We use BRITE [MLMB01] to generate the large network topology, which uses the statistics extracted from real network measurements. BRITE can produce random net-

Table 4.3: Statistics for Multiple Client Network with Four Flows (with 0 Delay Increment)

	TCP	RTCP
Flow 1's Throughput (Mb/s)	11.638	9.966
Flow 2's Throughput (Mb/s)	9.305	9.898
Flow 3's Throughput (Mb/s)	8.574	9.752
Flow 4's Throughput (Mb/s)	7.663	9.745
Aggregate Throughput (Mb/s)	37.18	39.361
Simulation Events	271,421	14,293
Exec Time Avg $\pm$ Std (sec)	1.354 ± 0.012	0.017 ± 0.0001
Event Ratio	1	0.053
Execution Ratio	1	0.012

Table 4.4: Statistics for Parking Lot Network

	TCP	RTCP
Per-Flow Throughput (Mb/s)	56.059	57.710
Aggregate Throughput (Mb/s)	224.237	230.839
Simulation Events	8,629,198	69,076
Exec Time Avg $\pm$ Std (sec)	33.991 ± 0.215	0.251 ± 0.002
Event Ratio	1	0.008
Execution Ratio	1	0.007

work models using a top-down method: it first generates a random network topology for the autonomous systems (ASes), then for each AS generates a router-level topology, and finally merges the AS-level and router-level topologies by connecting the routers belonging to different ASes. For the experiment, we use BRITE to generate a network topology consisting of 10 Autonomous Systems (ASes), each with 10 routers. We then randomly choose 2 routers in each AS and attach a single campus network to each selected router. The campus network is a synthetic network defined by [Nic02]. Each campus network has 13 LANs, 508 hosts with 504 clients, 4 servers, and 18 routers. The resulting topology consists of 10 ASes and 20 campus networks with a total of 460 routers and 10,160 hosts. To generate traffic, we randomly select the clients and have them download a 100 MB file from a randomly selected server. We schedule the clients to initiate downloads with an ex-

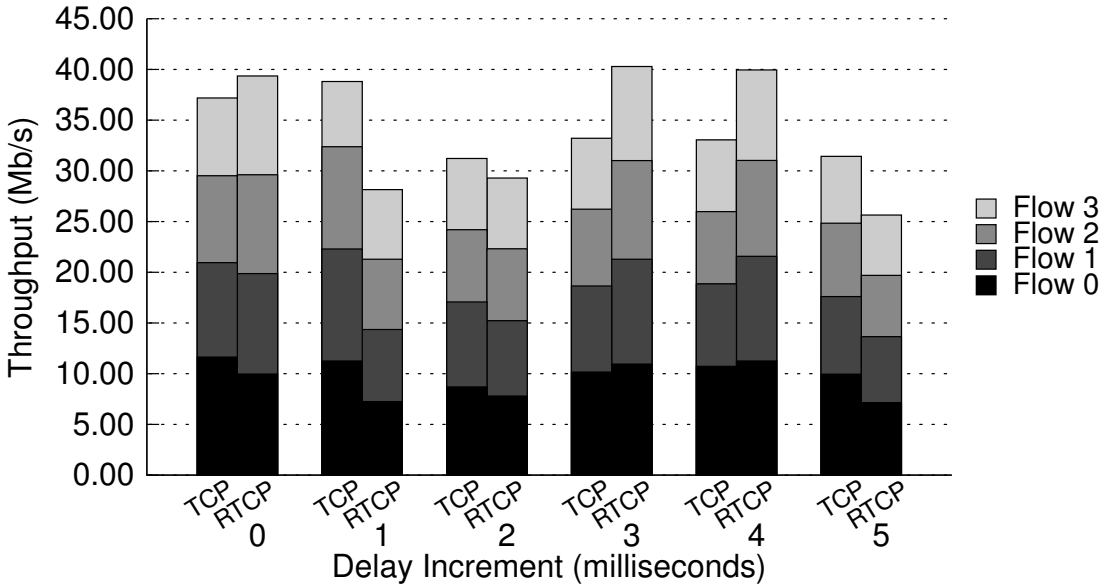


Figure 4.6: Throughput of the multiple client network

ponentially distributed inter-arrival time with a mean of 1 second. We run the simulation for 300 seconds and measure both the aggregate and average per-flow throughputs.

The results are shown in Table 4.5. The speedup achieved by RTCP is significant in this case. RTCP reduces the number of events by a factor of 60 and the execution time by a factor of 200. The aggregate throughput using RTCP, however, is 15% lower than that of TCP. Fig. 4.9 is the Q-Q plot for comparing the empirical distributions of the throughput for individual flows between RTCP and TCP. The curved pattern suggests that the central quantiles are more closely spaced in TCP throughput than in RTCP throughput. We see that overall the flows match quite well, although RTCP seems to have underestimated the throughput at the low end of the range and overestimated the throughput at the high end of the range.

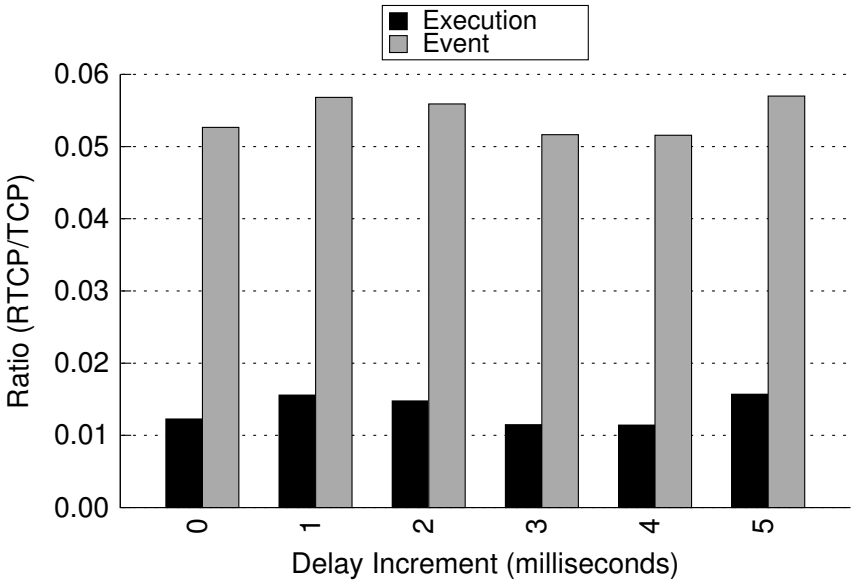


Figure 4.7: Reduction in execution time and number of events for the multiple client network

Table 4.5: Statistics for Large Network

	TCP	RTCP
Aggregate Throughput (Mb/s)	857.320	735.708
Simulation Events	433,019,043	7,038,100
Exec Time Avg $\pm$ Std (sec)	$3,700.955 \pm 336.036$	19.314 ± 1.371
Event Ratio	1	0.016
Execution Ratio	1	0.005

4.6.5 Discussion

RTCP is able to capture the behavior of TCP reasonably well. However, RTCP tends to underestimate throughput and the average queue size. We suspect that this is because the analytical models used by RTCP react to data losses faster than they should. Further, we observed that the analytical models seem to be less sensitive to amount of data loss. For small models, these errors cause negligible differences (less than 5.4%) in the overall throughput, which is confirmed by our experiments. For large models, however, the effect on the throughput is noticeable (14% difference). In the last experiment, RTCP signifi-

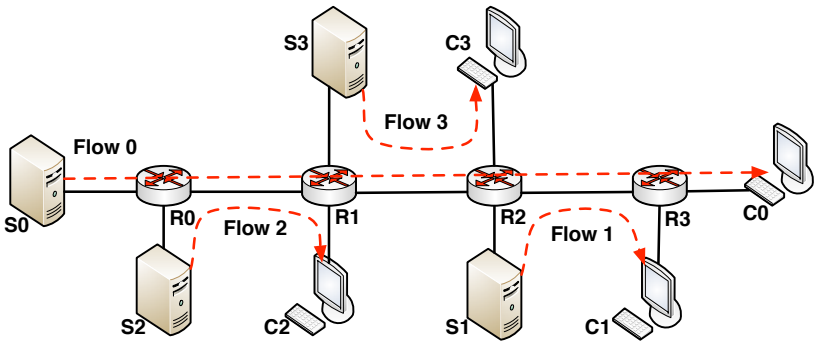


Figure 4.8: Parking lot network

cantly under estimates the throughput of a handful of flows that happen to contribute a large portion of the aggregate throughput. Besides of the inaccuracy introduced by the abstraction of the traffic at "rate window" level, the deviation can also be explained by the analytical TCP models we adopt.

The model proposed in [CSA00] has been evaluated through simulations, controlled Internet measurements and comparing with live traces under many different scenarios with different RTT, transfer size, MSS, and maximum congestion window size. The connection establishment model has been shown to have good accuracy as long as its assumption holds (i.e. the loss rate is below 0.5). The data transfer model that consists of the slow start model and the steady-state model presents variable performances of different lengths of data transmission under different loss conditions. Since the model proposed by [PFTK00] is used as an approximate model to deal with the data transfer in congestion avoidance model, the errors introduced here are mainly because of the two limitations of this approximation. First, in the approximate model, once the first loss is detected, the window size will be immediately adjusted to its steady state value. However, in real TCP implementation, when the sender detects a loss in the initial slow start phase, its instantaneous window size is often much larger than the steady-state average congestion window size [CSA00]. Therefore it takes a period of time for the sender to adjust its congestion

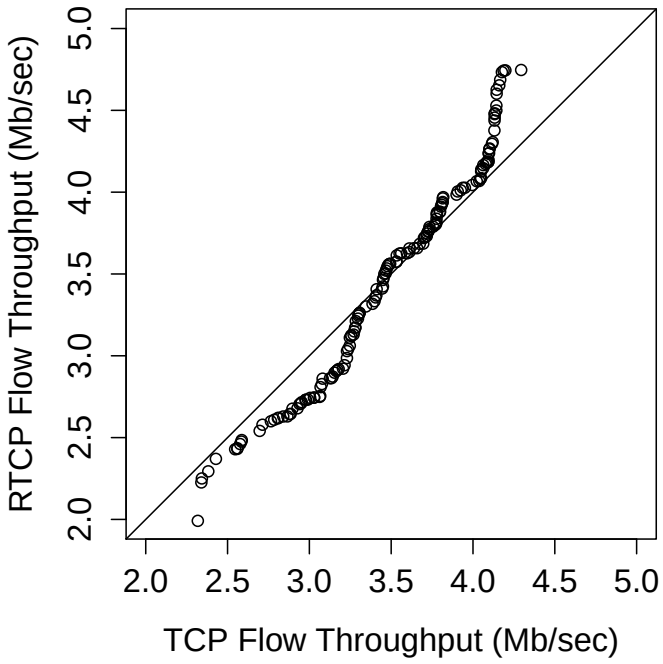


Figure 4.9: Q-Q plot of RTCP throughput versus TCP throughput for large network

window size from its value at that moment to its steady state value. The lower the loss rate is, the longer the duration of this transition time will be. According to the results shown by [CSA00], for high loss rates (i.e., 5% and higher), the sender exits slow start and immediately drops its window size to a small value that is close to the steady-state window value. So the error of the model in this case should be small. For low loss rates (i.e., 0.1% and below), it takes longer, which is usually three or more loss indications, to transit from slow start mode to steady-state phase. So the model in this case often over-estimates the transmission latency. Consequently, the throughput will be underestimated. Second, Padhye et al.'s model does not model slow start after retransmission timeouts. For loss rates above 1%, the send rate of the steady-state phase is similar to that of the slow start phase after retransmission timeouts. So the error of the model should be small.

On the other hand, for lower loss rates, although retransmission timeouts rarely happen, once they occur, the errors will be inevitable.

Regardless of these errors, both models we apply have been proved their accuracy for most cases. In addition, the implementation of our RTCP model allows the analytical sending models to be easily replaced with other solutions for different targets. For example, we assume TCP RENO because Padhye et al.'s model is restricted to model TCP RENO. However, by replacing with other TCP analytical models, our RTCP model can definitely provide solutions for modeling other TCP protocols.

Overall, for all cases, the estimated instantaneous queue size is similar between RTCP and TCP. More importantly, the inaccuracies are well compensated by the reduction in the number of events and in the execution time when the packet-level details are not critical for the goal of the study. For large experiments, we can observe a speedup of over two orders of magnitude.

4.7 Conclusion

This chapter presents a traffic model which reduces the time and space complexity for simulating TCP traffic behavior with good accuracy. The proposed RTCP model introduces a new level of granularity for representing the network traffic. Rather than modeling at the granularity of individual packets, we approximate the traffic flows as individual rate windows, each consisting of a number of packets with the same arrival rate. To model the detailed TCP behavior, we use existing analytical models to calculate the send rate at the traffic source as a function of the measured round-trip time and packet loss probability. We calculate the queuing delay and the loss probability as the rate window traverse individual network queues along the flow path. Experimental results are encouraging. They show that RTCP can speed up simulation of large network traffic by a factor of 200, while

still maintaining a reasonable level of accuracy. For all the experiments we conducted, RTCP model has been shown to achieve more than 50x speed-up over TCP model.

CHAPTER 5

CLUSTER-BASED SPATIO-TEMPORAL BACKGROUND TRAFFIC GENERATION

In this chapter, we propose a cluster-based spatio-temporal background traffic model that aims to produce network-wide traffic on arbitrary network topology while maintaining the spatial and temporal characteristics of the observed Internet traffic.

5.1 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.

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 based on traffic matrices. They focus specifically on aggregate traffic intensity (rather than individual flows or packets) and can only deal with variations at coarse time scales (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 the traffic traces collected from individual links; they generate traffic as individual flows or packets, and therefore can capture the effect on the target applications more accurately. Temporal models, however, can only work with individual links or paths; they are not able to capture the spatial distribution of traffic on 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 chapter, we aim to provide a spatio-temporal background traffic generator that can be easily applied for network studies. To this end, we posit the following criteria that the traffic generator should meet :

- *Spatio-temporal correlation*: The traffic generator should jointly consider both spatial and temporal structures; that is, the generator not only needs to reproduce the bursty traffic behavior as observed on individual links and/or paths, but also it needs to reasonably place traffic on the target network topology so that it can 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 changes over time.
- *Flexibility*: The traffic generator should be flexible: the generator needs to produce traffic with various traffic conditions, for different scenarios, and on arbitrary network topologies, in order to test the target applications. At the same time, it is important that the generated traffic maintains a good level of realism. That is, the traffic generator needs to provide the necessary "control knobs"—the ability to tune certain parameters—while keeping important spatial and temporal traffic characteristics invariant.

To derive the spatio-temporal background traffic model, we start by analyzing the traffic behavior observed from real network (i.e., from 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 multi-dimensional attributes and then apply a clustering algorithm to group the end hosts

as seen from the traffic trace. Different from the previous traffic clustering approaches, we carefully choose the features that define the clusters to facilitate traffic generation. The result traffic classes are then mapped onto a given network topology. This is achieved by first stochastically determining the 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 multi-dimensional 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. *Our method can thus capture both temporal and spatial structures from real Internet traffic observations.* In addition, we also provide a method to scale the traffic intensity level on the network links while maintaining the same spatial and temporal characteristics, and thus enable testing applications under various and yet realistic traffic conditions.

Our cluster-based spatio-temporal background traffic generation method has been validated through extensive simulation experiments. The results show that the generated traffic is statistically similar to the original traffic traces used for the traffic generation. The proposed method is not limited to network simulation; it can be applied for network emulation as well as in empirical studies. In this article, we focus only on the simulation aspects of the traffic generator.

Table 5.1: 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

5.2 Background

In this section we first describe the current state of using background traffic in network studies by conductiong a survey for papers in SIGCOMM 2007-2013. We then provide a brief summary of related work in traffic classification. A detailed review of existing traffic models, including temporal, spatial, and spatio-temporal models, can be found in Chapter 2.

5.2.1 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 appeared in the last seven years (2007-2013). We categorize the papers that involve experiments with infrastructural networks according to their evaluation methods, including real testbeds (R) , simulation (S), emulation (E), or a combination of them. Table 5.1 shows the results. The "other" category consists of work that does not involve experiments with infrastructure networks, such as wireless communications and pure 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 vari-

Table 5.2: Use of Synthetic Background Traffic in Some SIGCOMM Papers

Paper	**Network Topologies**	**Traffic Types**
probabilistic early response TCP [BRZL07]	simple topologies with single and multiple bottlenecks	long-lived TCP
service differentiation [PG08]	simple topologies with single and multiple bottlenecks	long-lived TCP sampled flows
scalable Ethernet architecture [KCR08]	campus network	trace playback
network measurement [PMH09]	simple topology with single bottleneck (dumbbell)	CBR traffic generator
network-wide redundancy elimination [ASA09]	Rocketfuel topologies	gravity model
Denial of service [LYX10]	simple topologies with single and multiple bottlenecks	long-lived TCP sampled flows
flow-level measurement [LDK10]	simple topology with single bottleneck (dumbbell)	sampled flows traffic generator
route reconfiguration [WWM$^+$10]	RocketFuel topologies US-ISP, GT-ITM, Abilene	gravity model trace playback
protocol manipulation attacks [KMM$^+$11]	simple topology with single bottleneck (dumbell)	long-lived TCP
flexible transport protocol [HGAS13]	simple topologies with single and multiple bottlenecks	sampled flows

ous network conditions for more flexibility, 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. Yet another common scenario is that the researchers use a real testbed to conduct small-scale studies, and then resort to simulation for large-scale experiments.

Next, we focus on simulation studies that require network traffic for evaluation. Table 5.2 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 inter-arrival times and flow lengths. Only a few studies involve existing traffic generators. All these studies are limited to simple network topologies, such as dumbbell.

For studies that require more realistic network topologies (such as using the Rocket-Fuel ISP PoP-level topologies [SMWA04]), people obtain traffic matrices generated from using simple assumptions or using the gravity model (e.g., [WWM+10, ASA09]). Anand et al. applied gravity model to estimate traffic matrix at PoP-level [ASA09], however, they assume that the traffic is uniformly distributed over the access routers within each PoP.

Wang et al. actually used the CAIDA trace to reproduce the packet-level traffic flows [WWM+10]. However, the derived temporal behavior of such traffic is independent of the spatial distribution; i.e., it does not preserve the spatio-temporal correlation. From this survey, we can conclude that there is significant lacking in the use of good network-scale background traffic models in experimental studies.

5.2.2 Traffic Classification

Our method uses clustering techniques to characterize traffic. There has been prior 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., [RSSD04] and [MHL+04]) 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., [KPF05] and [XlZB05]) classify traffic based only on end-user behavior but remain indifferent to network dynamics (such as network congestion and delays). The third type of methods (e.g., [WMK06]) 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 proposed a traffic content generative mode [VK12] that uses clustering techniques to determine the role of end hosts (as 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.

5.3 Overview of Cluster-Based Spatio-Temporal Traffic Generation

This section presents an overview of our cluster-based spatio-temporal traffic generation method. The method makes the following assumptions:

- Our 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 our 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 could implement methods to intentionally and probabilistically create traffic load imbalance. Again, we will explore this issue in our future work.

Our method can be divided into three steps:

1. We first analyze the traffic trace, by characterizing the traffic as a function of multi-dimensional 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 with large data volume) and heavily connected servers. The high-level traffic behavior is then summarized by the 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 then assign the clusters (from the previous step) to the routers. Because we are interested in the interaction between the foreground and background traffic at the network links connecting the routers, the background traffic generator only needs to produce traffic on those links between routers, as opposed to modeling the individual end hosts. Each router presumably can connect to many end hosts, and each end host would belong to a cluster. That is, a router may contain multiple clusters. The goal of this step is to proportionally distribute the clusters over routers of an arbitrary network topology.
3. With 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 clustering results.

In the following three sections, we present the details of the three steps of our proposed method, respectively.

5.4 Step 1: Traffic Classification

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

5.4.1 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 points. They include a CAIDA trace, collected from a US backbone link [CAIb]; a MAWI trace, collected from a trans-Pacific link [MAW]; and a campus network trace, collected at the uplink from the University of Napoli [DPR+08, DPV09]. 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 150 Mbps 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' 200 Mbps uplink (connecting the campus network to the rest of the Internet).

For the traffic analysis shown below, 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). Figure 5.1 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

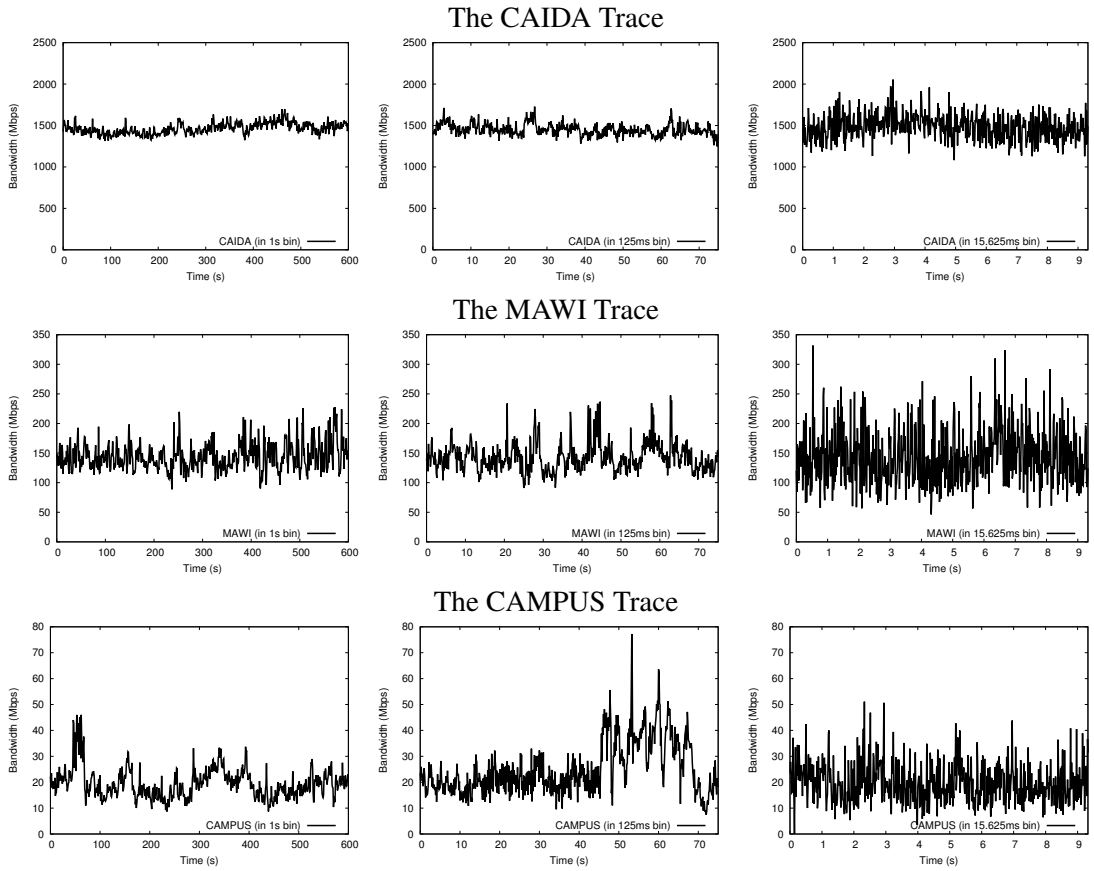


Figure 5.1: Traffic intensity for the CAIDA, MAWI, and CAMPUS traces.

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 figure shows intuitively the scale-free traffic behavior by simply zooming in, i.e., by progressively increasing the time resolution.

5.4.2 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, source and destination IP addresses, and the TCP flags from the packet headers. These attributes can reveal the hidden correlation of the spatial distribution underpinning the user access patterns, as well as the temporal behavior of individual traffic flows.

We conduct k-means clustering [Mac67], in particular, using a data mining software, called WEKA [HFH+09]. 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 with 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 where the components within each cluster behave similarly. We note that the values of some of the attributes may differ in magnitudes. For example, the number of flows involving a particular IP address ranges from one to several hundreds; the total size of data sent or received by an IP can vary from zero to several megabytes or more. For these attributes, we take logarithm of the values for clustering.

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. Choosing k should balance between accuracy and performance. 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. Figure 5.2 shows the clustering errors for all three traces with different values of k.

Table 5.3 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);

Table 5.3: 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

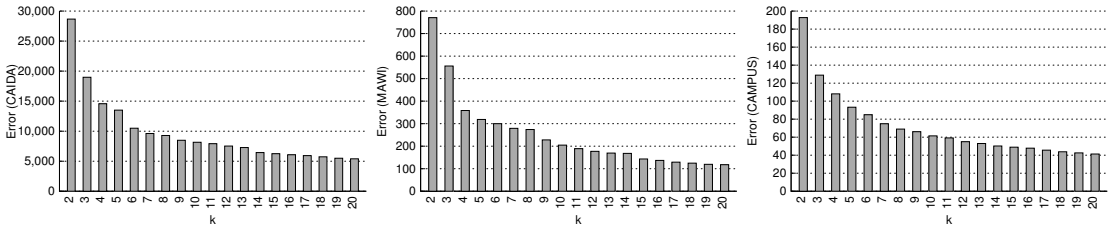


Figure 5.2: Clustering error of different k values for the CAIDA, MAWI, and CAMPUS traces.

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, i.e., 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.

5.4.3 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 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 *population density* for cluster i as:

$$\phi_i = \frac{|C_i|}{\sum_{0 \leq j < k} |C_j|} \tag{5.1}$$

- 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 $0 \leq i, j < k$. Note that for simplicity, we

do not distinguish the direction of the flows. We count each flow in both directions: a flow 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:

$$F = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} F_{ij} \tag{5.2}$$

We then calculate the *flow density* for cluster i as the proportion of flows that involve an IP address in cluster i among all flows:

$$\psi_i = \frac{\sum_{0 \le j < k} F_{ij}}{F} \tag{5.3}$$

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:

$$\omega_{ij} = \frac{F_{ij}}{\sum_{0 \le x < k} F_{ix}} \tag{5.4}$$

- 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:

$$\lambda_{ij} = \frac{F_{ij}}{T} \tag{5.5}$$

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.

- 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 lognormal distribution:

$$\mu = \ln\left(\frac{m^2}{\sqrt{v+m^2}}\right) \tag{5.6}$$

$$\sigma = \sqrt{\ln\left(1+\frac{v}{m^2}\right)} \tag{5.7}$$

- We calculate the aggregate traffic rate 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:

$$\delta_{ij} = \lambda_{ij} \cdot e^{\mu_{ij}+\frac{\sigma_{ij}^2}{2}} \tag{5.8}$$

The total in-flow rate and out-flow rate at cluster i, also in number of bytes per second, can be summed up easily:

$$\delta_{\bullet i} = \sum_{j=0}^{k-1} \delta_{ji} \tag{5.9}$$

$$\delta_{i\bullet} = \sum_{j=0}^{k-1} \delta_{ij} \tag{5.10}$$

Note that both in-flow rate and out-flow rate include traffic going between end hosts in the same cluster.

Lognormal distribution is appropriate for describing the flow size. Figure 5.3 shows the Q-Q plots of the flow size between two selected clusters from the CAIDA, MAWI, and CAMPUS traces independently against a lognormal distribution with parameters estimated from the trace data. They match well. We observe similar results with all other clusters for the three traces, although they use different lognormal parameters.

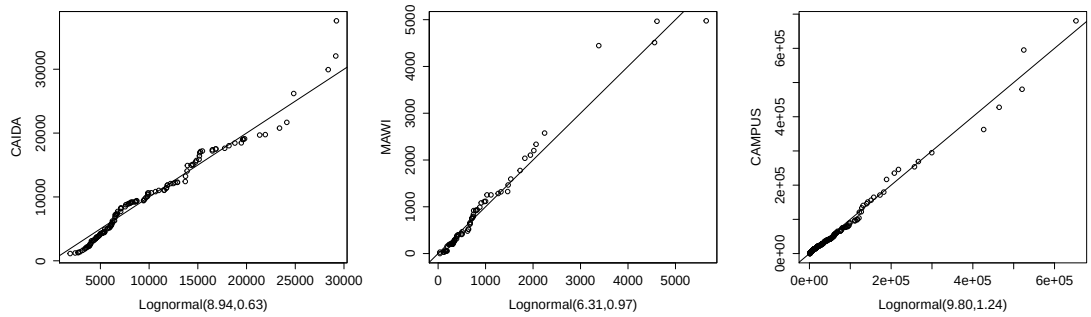


Figure 5.3: Q-Q plot of flow size vs. lognormal for the CAIDA, MAWI, and CAMPUS traces.

5.5 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 to produce the traffic accordingly. However, this would be unnecessary since we are only interested in generating the background traffic on the links between routers of the given topology. That is, one can simply have the routers to assume the role of traffic sources and destinations and produce the traffic amongst them. Since each router may conduct traffic that belong 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 is mostly proprietary and thus would be 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 based on "common sense". We assume that the traffic load on any link should not exceed its link capacity; also, the traffic should be distributed over the network evenly so that it would not load any particularly link disproportionally.

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 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.

5.5.1 Deriving Traffic Matrix for Arbitrary Network Topology

We first derive the traffic matrix for any arbitrary topology. Suppose the network consists of n routers and m links. The goal is to calculate the traffic rate r_{ij} from router i to router j, for all $0 \leq i, j < n$ and $i \neq j$. We adopt the method proposed by [NST05] 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 [NST05].

Sampling Random Flow Rates We assume that the flow rates observe the lognormal distribution. It has been shown that lognormal distribution provides the best fit for both Sprint and Abilene networks [NST05]. For a given network topology and traffic routing (again, we assume static routing), we first determine the lognormal parameters μ and σ.

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:

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

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 expect the mean of the lognormal 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 stardard deviation by the same factor, we expect it to be $\sigma_c \rho \gamma^{-1}$. Therefore, we can calculate the parameters for the lognormal distribution, following Equations (5.6) and (5.7):

$$\begin{aligned} \mu &= \ln(\mu_c^2 \rho/\gamma) - \frac{1}{2}\ln(\mu_c^2 + \sigma_c^2) \\ \sigma &= \sqrt{\ln(1 + \sigma_c^2/\mu_c^2)} \end{aligned}$$

Given μ and σ, we take a sample of size $n(n-1)$ from the lognormal distribution; we denote the sampled flow rates as $A_0, A_1, \ldots, A_{n(n-1)-1}$.

Integer Linear Programming (ILP) Next, we simply follow the the method proposed by [NST05], 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 maximal link utilization should be the maximum of all link utilizations. The link utilization is defined as the traffic rate contributed by all OD flows traversing this link divided by the link capacity. The problem is

formulated as the follows:

Minimize: z_{max}

Subject to:

$$z_{max} \geq z_{uv} = \sum_{0 \leq i,j < n, i \neq j} \frac{x_{uv}^{ij}}{C_{uv}}, \ \forall L_{uv} \in E \tag{5.11}$$

$$x_{uv}^{ij} = \begin{cases} \xi_{ij} & \text{if } L_{uv} \in P_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

$$\xi_{ij} = \sum_{p=1}^{n \times (n-1)} I_p^{ij} A_p \tag{5.13}$$

$$\sum_{0 \leq i,j < n, i \neq j} I_p^{ij} = 1 \tag{5.14}$$

$$\sum_{p=1}^{n \times (n-1)} I_p^{ij} = 1 \tag{5.15}$$

$$\sum_{u:L_{uv} \in E} x_{uv}^{ij} - \sum_{v:L_{uv} \in E} x_{uv}^{ij} = \begin{cases} \xi_{ij} & \text{if } u \text{ is source} \\ -\xi_{ij} & \text{if } u \text{ is destination} \\ 0 & \text{otherwise} \end{cases} \tag{5.16}$$

Equation 5.11 defines the maximum link utilization z_{max}, where C_{uv} is the capacity of the link L_{uv}, and x_{uv}^{ij} is the traffic rate contributed by the OD-pair from router i to j on the link L_{uv}, which is defined in Equation 5.12. Equation 5.13 defines the rate of each OD-pair ξ_{ij} by using the mapping indicator I_p^{ij}, where $0 \leq p < n(n-1)$, $0 \leq i,j < n$ and $i \neq j$.. The indicator is set to be 1 if flow rate A_p is assigned to the source-destination pair from router i to router j, and 0 otherwise. Equation 5.14 and 5.15 guarantee that each traffic rate A_p is mapped to exactly one OD-pair ξ_{ij} and vice versa. Equation 5.15 is the flow conservation equations. The output of the solution is a set of mapping indicators I_p^{ij}.

Consequently, we can obtain the traffic matrix, where

$$r_{ij} = \begin{cases} \sum_{p=0}^{n(n-1)-1} I_p^{ij} A_p & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases}$$

5.5.2 Solving Cluster-to-Router Mapping

We have obtained the cluster-level traffic summary from traffic classification (in step 1). Following the previous section, we have also obtained a traffic matrix, which contains the traffic demand, r_{ij}, from any router i to any other router j in the given network topology. In this section, we present an algorithm to assign the clusters to routers. More specifically, we solve for p_{si}, which is the proportion the end hosts belonging to cluster s are mapped to router i, where $0 \leq s < k$ and $0 \leq i < n$.

For a given network topology, we define the *user density* of router i as d_i, where $0 \leq d_i \leq 1$ and $\sum_{0 \leq i < n} d_i = 1$. A router's user density is a user-defined value; it shall 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: $d_i = 1/n$.

We note that there can be discrepancy 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:

$$\theta = \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} r_{ij}}{\sum_{s=0}^{k-1} \sum_{t=0}^{k-1} \delta_{st}} \tag{5.17}$$

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 mapping the end hosts of cluster s to router i, so that we can maximize the spread of the traffic, i.e., make the clusters distributed as evenly as possible, among all routers, subject to the constraints dictated by the given traffic matrix. It is very unlikely to have a universally best objective function due to the complexity of the network traffic and the limitation of the traces publicly available. Our goal is based on the assumption that the packet trace observed from individual links is representative and can reveal the network-wide traffic behavior. The optimization problem can be formally specified as follows:

Minimize:

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

Subject to:

$$\sum_{i=0}^{n-1} p_{si} = 1, \quad \forall s \in \{0,1,\ldots,k-1\} \tag{5.18}$$

$$p_{si} \geq 0, \quad \forall s \in \{0,1,\ldots,k-1\}, i \in \{0,1,\ldots,n-1\} \tag{5.19}$$

$$\sum_{s=0}^{k-1} p_{si} \cdot \phi_s = d_i, \quad \forall i \in \{0,1,\ldots,n-1\} \tag{5.20}$$

$$\theta \sum_{s=0}^{k-1} p_{si}\delta_{s\bullet} - \sum_{j=0}^{n-1} r_{ij} = \theta \sum_{s=0}^{k-1} p_{si}\delta_{\bullet s} - \sum_{j=0}^{n-1} r_{ji}, \quad \forall i \in \{0,1,\ldots,n-1\} \tag{5.21}$$

$$\theta \sum_{s=0}^{k-1} p_{si}\delta_{s\bullet} \geq \sum_{j=0}^{n-1} r_{ij}, \quad \forall i \in \{0,1,\ldots,n-1\} \tag{5.22}$$

Equation (5.18) states that the proportion of mapping the end hosts of cluster s to all routers should sum up to 1. Since they are all proportions, Equation (5.19) states that they should not be negative. Equation (5.20) defines the user density at router i, which is the sum of the proportion of end hosts of each cluster mapped to router i, multiplied by the cluster's population density.

Equation (5.21) matches the flow rates observed by the cluster-level traffic summary with those specified by the traffic matrix at each router. The first term on the left-hand side of the equation is the sum of the out-flows 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 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-flows of all clusters assigned to the router and the total traffic received by the router as seen by traffic matrix, which is also the traffic between the end hosts attached to the same router. Equation (5.22) makes sure that the difference shall not be negative.

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

5.6 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 in the following steps:

1. We model the flow arrivals as a Poisson processes (using exponentially distributed inter-arrival time), with an arrival rate:

$$\lambda = \frac{\alpha \mathsf{F}}{\mathsf{T}} \tag{5.23}$$

   where F is the total number of flows observed in the packet trace (Equation 5.2) and T is the duration of the trace. α is a scaling factor; it is a user-defined "control knob" for varying the traffic intensity 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 the probability equal to the cluster's flow density, ψ_{s} (Equation 5.3).

3. Select the source router i with probability p_{si}, which is the proportion of mapping the end hosts of cluster s to router i.

4. Select the destination cluster t from cluster s using the peering probability, ω_{st} (Equation 5.4).

5. Select the destination router j with probability p_{tj}, which is the proportion of mapping the end hosts of cluster t to router j.

6. We create a TCP flow from router i to router j and transfer data of a certain amount; we sample the flow size in the number of bytes from the lognormal 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, for simple packet inspection[1]. 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 Nd_i IP addresses, where d_i is the user density of router i (see previous section). We can assign these IP address to clusters based on the cluster distribution at this router. In particular, one can assign the IP addresses associated with router i to cluster c using the following proportion:

$$\kappa_{ci} = \frac{\mathsf{p}_{\mathsf{ci}} \cdot \phi_{\mathsf{c}}}{\sum_{\mathsf{s}=0}^{\mathsf{k}-1} \mathsf{p}_{\mathsf{si}} \cdot \phi_{\mathsf{s}}}$$

where p_{si} is the proportion the end hosts belonging to cluster s are mapped to router i (resulted from the previous step), and ϕ_{s} is the population density for cluster s (see Section 5.4.3). After we associate the IP addresses to the routers, we can now generate the flows with specific source and destination IP addresses. More specifically, in step (3), we can select the source address from all hosts attached to router i that belong to cluster s uniformly at random. Similar, in step (5), we can select the destination address from all hosts attached to router j that belong to cluster t again uniformly at random.

5.7 Experiments

In this section we validate our spatio-temporal background traffic model, particularly focusing on the aspects of spatio-temporal correlation, realism, and flexibility. We conduct the experiments under two scenarios. First, we use a real backbone network, the Abilene network, to evaluate the basic properties of the generated traffic, including the traffic constitution and its spatial distribution. Second, we use a synthetic campus network to investigate the temporal characteristics of the generated traffic on different links of the

[1] 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 properly distributed over the entire network.

Figure 5.4: The Abilene network.

network, including the variations of the traffic intensity and the traffic burstiness. In addition, we study the spatial distribution of the generated traffic and its effect on the behavior of foreground traffic with different scaling factors.

5.7.1 The Abilene Network

The Abilene network, as shown in Fig. 5.4, contains twelve routers and fifteen links. The link connecting Indianapolis and Atlanta has a bandwidth of 2.5 Gbps and all the other links have a bandwidth of 10 Gbps. 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. [ZGGR05]. For packet trace, we use the CAIDA trace collected from a US backbone link, as described in Section 5.4. In this experiment, since we only focus on the constitution and distribution of the background traffic, and since there is no foreground traffic to interact with the generated background traffic, we simplify the traffic generator by replacing the computationally expensive packet-oriented simulation with a fluid model with constant rate flows (in step 6

of the traffic generation algorithm in Section 5.6) for expediency. We record the generated traffic on all network links of the network for analysis.

First, we examine whether the generated traffic would constitute the same type of flows as in the original packet trace. This can be achieved by analyzing the generated traffic trace, applying same clustering algorithm with the same set of attributes, and comparing clustering results between the original packet trace and the generated traffic trace. In particular, we use a commonly used metric, called "rand index", to determine clustering similarity. For each generated traffic trace (one for each link), let N be the total number of observations (i.e., distinct IP addresses) appeared in the trace (there are $N(N-1)/2$ observation pairs). If the pair of IP addresses are from the same cluster, we say they are co-members. Let N_{11} be the number of observation pairs that are co-members and still remain to be co-members when the generated traffic trace is clustered again. On the contrary, let N_{00} be the number of observation pairs which are not co-members and still belong to different clusters when the generated traffic trace is clustered again. We define the rand index, S_R, as follows:

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

Fig. 5.5 shows the rand index for every link of the network. We see that the value never goes below 0.75, which means the majority of the IP addresses maintain similar membership association as in the original classification. The constitution of the traffic matches well with the cluster assignment over the entire network when the traffic is spread among all links of given network topology.

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 overall traffic distribution (i.e., the percentage of traffic on the links) of our spatio-temporal method (Optimal) against those from the original traffic matrix (Real). To make

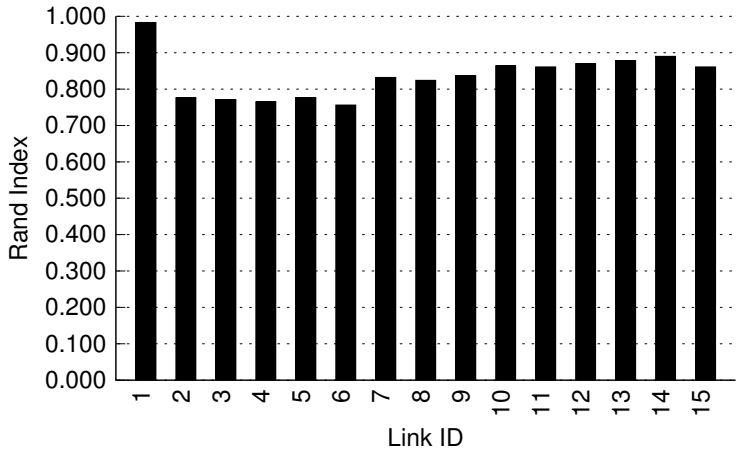


Figure 5.5: Rand index of all links of the Abilene network.

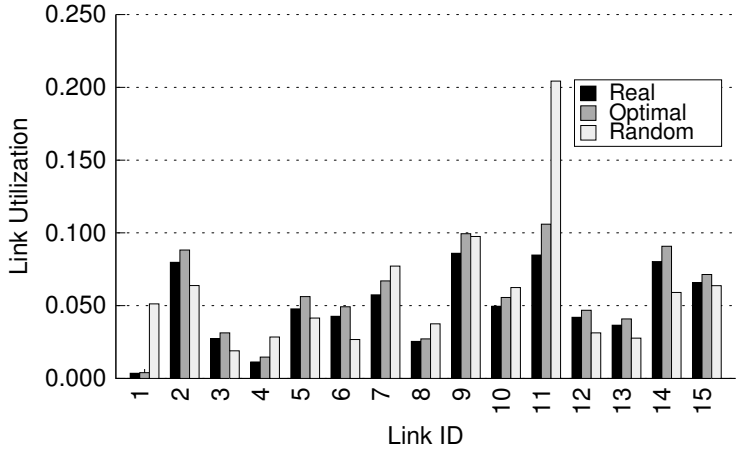


Figure 5.6: Utilization of all links of the Abilene network.

it more interesting, we also compare it with the results from using a method that places the flows over the network uniformly at random (Random). Fig. 5.6 shows the link utilization on all fifteen links, and Fig. 5.7 shows the percentage of traffic on each of the fifteen links. The results demonstrate that our spatio-temporal method (Optimal) is able to accurately represent the spatial distribution of the traffic as shown by the real traffic matrix.

In addition, we compare the flow sizes distribution between the generated traffic and the measurement. Fig. 5.8 shows the Q-Q plots of the flow sizes between the generated

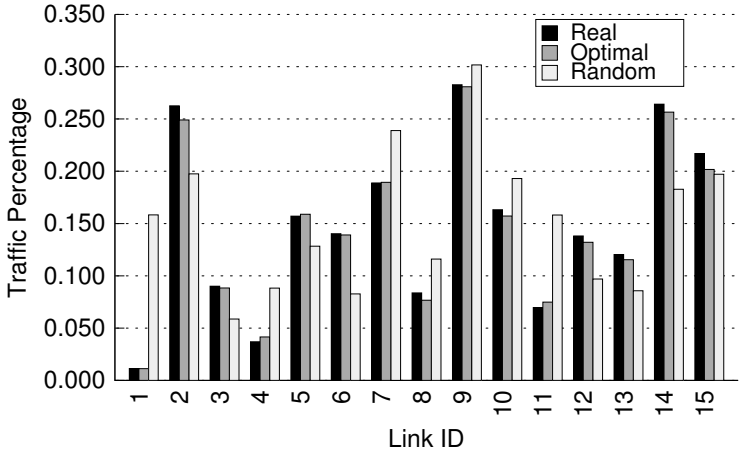


Figure 5.7: Traffic distribution among all links of the Abilene network.

traffic we measured from all links of the Abilene network and the empirical distribution extracted from the trace. We can see that although the flow size distributions of all links are similar to the original distribution.

5.7.2 The Campus Network

In the following experiment we use the packet-level detailed TCP as transport so that we can investigate the temporal characteristics of the generated traffic on different links of the network. We use a synthetic campus network topology, as shown in Fig. 5.9, 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. To make it more interesting, we designate different bandwidths to links; varying the link capacity allows us to observe the non-uniform traffic distribution over the entire network. For packet trace, we use the CAMPUS trace collected at a university campus unlink, as described in Section 5.4.

We examine the variation of intensity of the generated traffic over time on all links of the campus network. In the experiment, we vary the scaling factor α to be 0.5, 1, 1.5, 2, 2.5, and 3, for different background traffic intensity levels. Fig. 5.10 shows the traffic

intensity on a 50-Mbps link (we observe similar results for the other links as well). Each row shows the results of a different scaling factor. Similar to plots in Fig. 5.1, 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 time scales, i.e., it remains scale free, regardless of the scaling factor.

To gain a more precise review of traffic burstiness, we use the wavelet-scaling plot, also known as the energy plot, which captures the correlations in the amount of traffic arrived at consecutive time intervals of a given size [FGHW99]. Fig. 5.11 shows the energy plot of the generated traffic on the same 50-Mbps link at different time scales and with different scaling factors. For comparison, we also plot the energy of the original CAMPUS trace. The x-axis shows a range of time scales, each being 2^{-j}, from $j = 0$ (one second) down to $j = 11$ (around 0.5 ms). 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 time scales. Furthermore, the generated traffic exhibits very similar burstiness when compared to the original packet trace, except when j is around 9 (about 2 ms), which is actually at a time scale not so much larger than the packet transmission time. A slight deviation in the traffic burstiness at this small time scale would have little impact on the foreground traffic; therefore, our traffic generator uses a fixed packet size when generating the TCP flows.

While the scaling factor has little effect on the traffic burstiness, it has significant impact on the traffic intensity, as one can see in Fig. 5.10. In Fig. 5.12, we show the link

Table 5.4: Link Utilization and Traffic Distribution Summary

α	link utilization		traffic distribution	
	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

utilization with different scaling factors, for all links that have background traffic[2]. We see that the traffic intensity increases almost proportionally on all links. Fig. 5.13 shows the traffic distribution (the percentage of traffic on the links). As expected, different scaling factors have little effect on the traffic distribution. Table 5.4 shows the mean and standard deviation of the link utilization and 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 [VV08] show that background traffic can have significant impact on applications, such as Web downloads, multimedia video streaming, and bandwidth estimation tools. Here, we simply perform an experiment to show that our generated background traffic can significantly influence the foreground applications. More specifically, we select two routers in the campus network (one 50-Mbps network and another in the 20-Mbps network) and have them transfer a 100 MB file using TCP. We repeat the experiment 25 times for each scaling factor. Fig. 5.14 shows the cumulative distribution function of the measured throughput. As expected, we see that the file transfer throughput decreases as expected when the background traffic intensity increases with larger scaling factor.

[2]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 and routed based on shortest-path.

5.8 Conclusion

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 multi-dimensional features, and then maps the traffic classes onto a given network topology for traffic generation. We validate our method both on a real backbone network and on a synthetic campus network. The results show that the model is able to generate representative background traffic with important spatial and temporal characteristics.

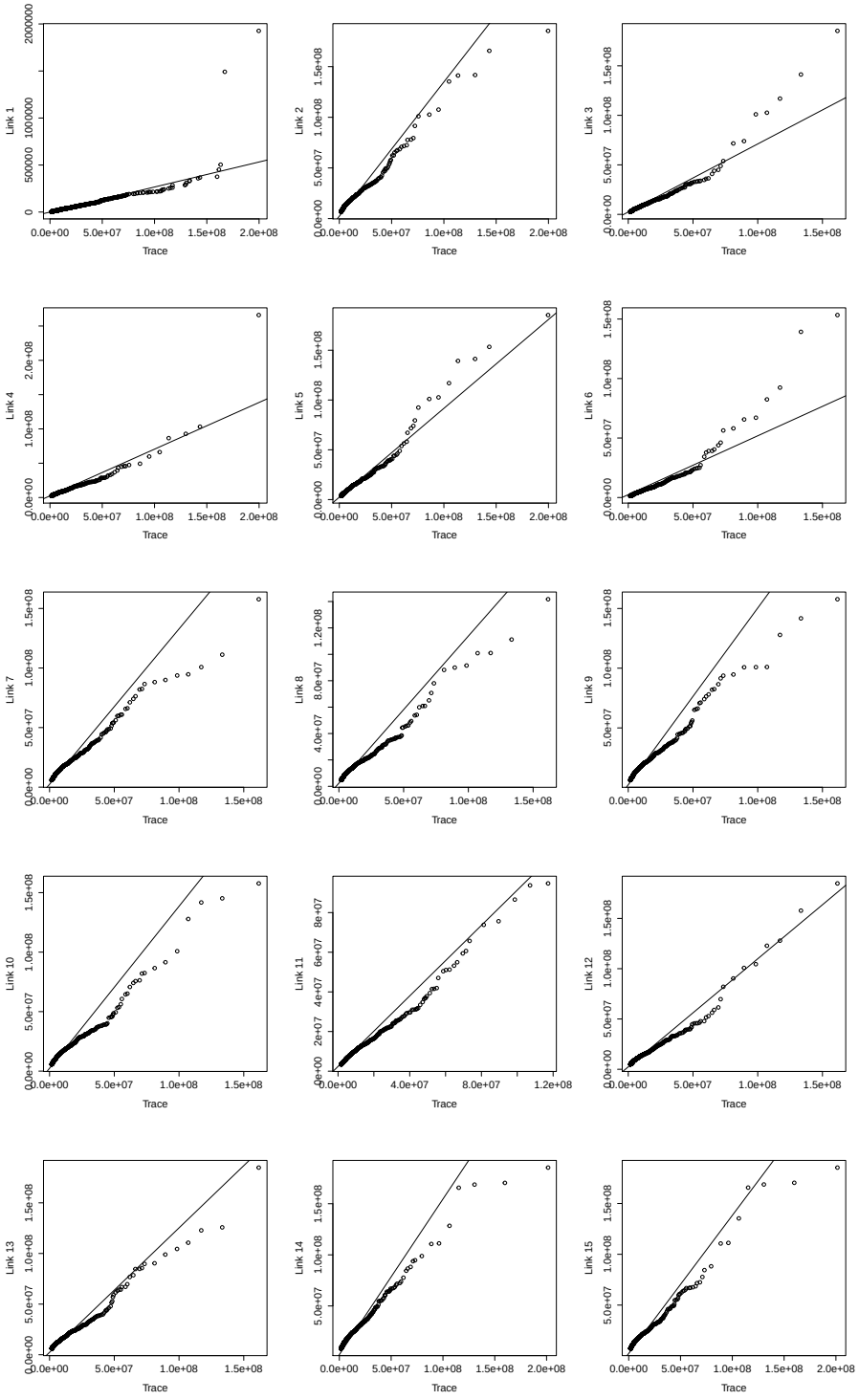


Figure 5.8: The Q-Q plots of the flow sizes between the generated traffic of all links of the Abilene network versus the flow sizes of the trace.

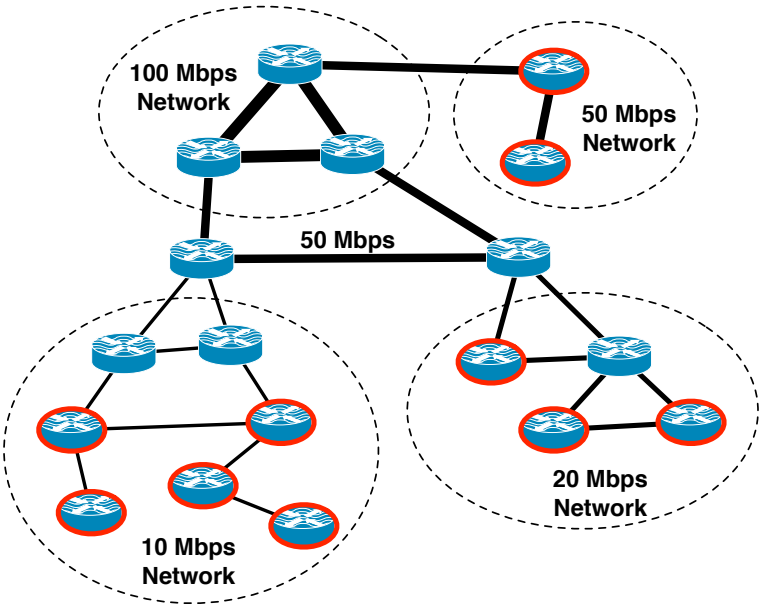


Figure 5.9: A synthetic campus network.

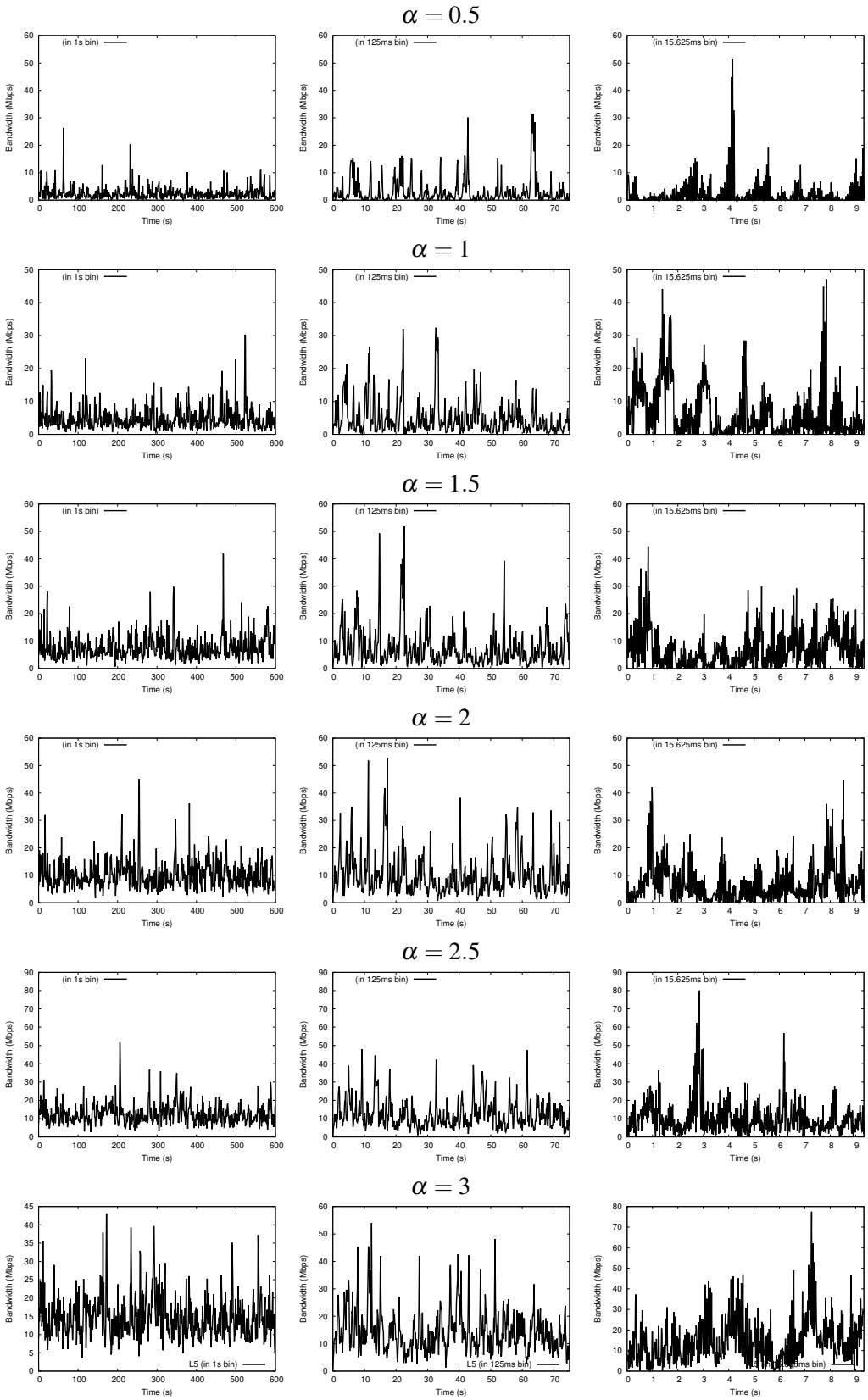


Figure 5.10: Traffic intensity for one link of Campus network with different scaling factors.

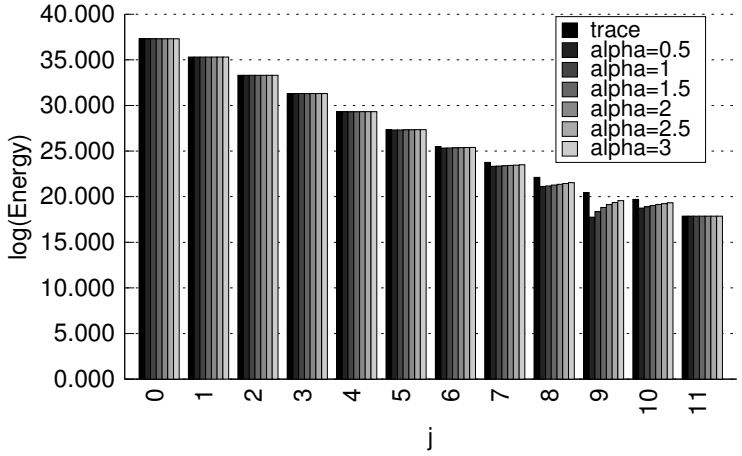


Figure 5.11: Energy plot of the CAMPUS trace and the generated traffic on the same link.

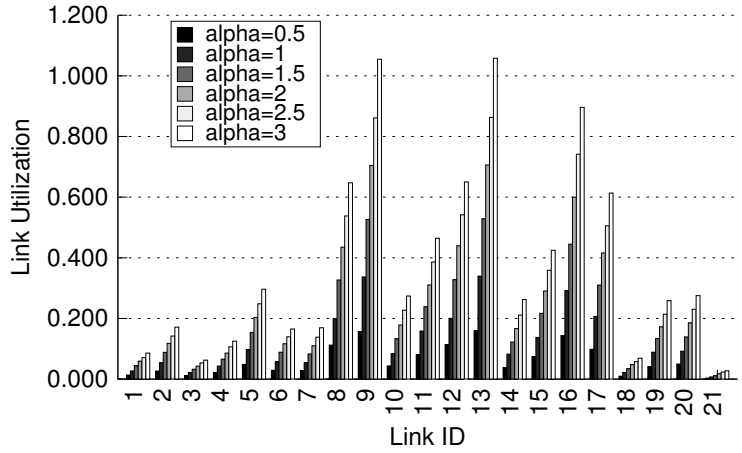


Figure 5.12: The link utilization of all links of Campus network with different scale factors.

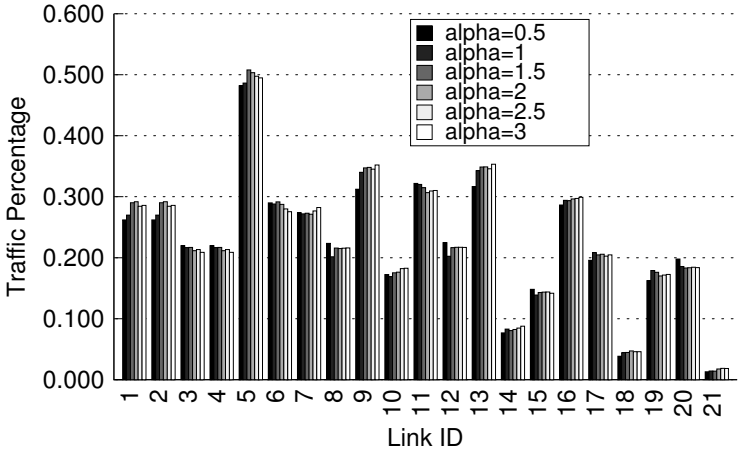


Figure 5.13: The traffic distribution over all links of Campus network with different scale factors.

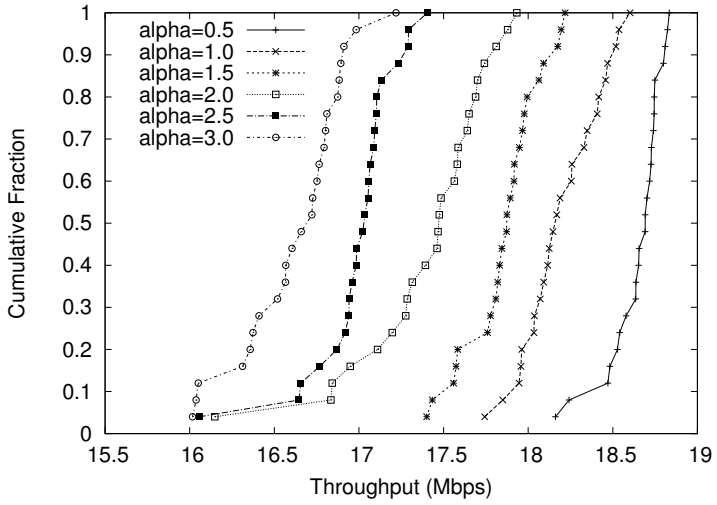


Figure 5.14: CDF of throughput of TCP downloads with different scale factors.

CHAPTER 6

CONCLUSIONS

This chapter presents a brief summary of this dissertation and a few possible directions for future work.

6.1 Summary

This dissertation focuses on generating representative background traffic for network experimentations. Specifically, this work addressed the following problems:

- The starting point of this work is an existing hybrid background traffic model. Artificial assumptions of the traffic model may lead to incorrect evaluation results. This work enhanced the existing model by removing its two unrealistic assumptions. One improvement is modeling ACK traffic so that the network condition on the reverse direction of the data traffic can be correctly reflected. The other improvement is integrating a PPBP (Poisson Pareto Burst Process) model to describe the traffic sessions so that the traffic burstiness can be reproduced. The improved model reflects the essential traffic characteristics without sacrificing efficiency.
- Scalability is another important issue background traffic modeling needs to address. Realizing that the existing background traffic models are either not scalable or loose too much accuracy in order to gain better performance, this work presented a fast rate-based TCP (RTCP) traffic model, the distinct feature of which is that the TCP congestion control behavior is represented using analytical models. This model outperforms other existing abstract traffic models in that it can correctly capture the overall TCP behavior and achieve a speedup of more than two orders of magnitude over corresponding detailed packet-oriented simulation.

- This work proposed a spatio-temporal traffic model that can be used for generating background traffic on the entire network for accurately evaluating applications and protocols in network studies. The proposed model incorporates clustering techniques to group traffic with similar characteristics into a few classes, and maps the traffic classes to access routers by solving a set of optimization problems. Thus the routers can generate traffic according to the spatial and temporal features each cluster represents. We believe that this work will fill the gap of the representative spatio-temporal background traffic models in the network research community.

6.2 Future Directions

This work can be extended in a few directions. We emphasize the extension of the spatio-temporal background traffic model because it addresses both spatial and temporal issues faced by background traffic modeling.

Immediate future work of the hybrid background traffic model proposed in Chapter 3 includes evaluating the model using real traffic traces. This model can also be extended to handle simultaneous TCP sessions with different round-trip times within the same fluid class.

As for the RTCP model, although we only evaluate RTCP as a pure rate-based traffic model in this work, the model can actually be extended to deal with mixed traffic generated by packet-oriented simulations. Nicol and Yan proposed a method for mixing packets and fluid flows [NY04], which can also be used with RTCP. We outline this method in the following. We keep track of the recent packet arrival rate. Once a packet arrives, we can estimate the current queue length by using the information of the aggregate arrival rate of packets and rate windows. Suppose a new packet of size S arrives at the queue at time t. The queue length can be updated as follows:

$$q(t) = [q_{last} + (t - t_{last}) * (\lambda_{rate} + \lambda_{pkt} - \mu)]_0^B$$

where λ_{rate} is the aggregate arrival rate of all rate windows, λ_{pkt} is the recent packet arrival rate, $\mathsf{t}_{\mathsf{last}}$ is arrival time of the previous packet, $\mathsf{q}_{\mathsf{last}}$ is the queue length at $\mathsf{t}_{\mathsf{last}}$. Whether we enqueue the packet depends on the current queue length, the mixture of different types of flows, and the queue capacity.

There are several things that we would like to explore with the spatio-temporal traffic generation model in the future work. Immediate future work includes the following:

- First, our traffic generation method is based on network measurements, assuming that the user behavior observed from one packet trace is representative and can reveal the network-wide traffic pattern. It is not enough 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 a network) 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 have to identify and deal with the correlations among the different traces. It is important to represent the idiosyncratic nature of different links at different locations for more realistic network-wide background traffic generation.
- Second, we would like to extend the method and introduce more control knobs 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.

- Third, we need to investigate the performance of the spatio-temporal model with different transport models. We used a detailed TCP and an abstract fluid model in the validation. This provided evidence that our spatio-temporal model is capable of combining with different underlying protocols. Our model aims to capture the user behavior at the application level. It is valuable to explore the tradeoff between accuracy and performance by using various traffic models that describe the flows at lower level, such as the detailed TCP model, the RTCP model, and other fluid models. Overall, one needs to accurately and efficiently capture the mutual interaction between the foreground applications and background traffic at the proper time scale.

- Last, but not least, we would like to extend our spatio-temporal method 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 spatio-temporal 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 spatio-temporally correlated. Content generation heavily depends on the specific application or the goal of the study, and consequently tends toward an ad-hoc process. For example, web studies generate traffic with web content characteristics, such as content freshness times and content reusability across time and linked documents [SCK03]; while security studies focus on generating traffic with different rules to model the benign and malicious traffic [VK11]. The resulted spatio-temporal content generative model will be useful for evaluating many applications, especially the ones emerged into the Internet recently. We list a few of them in the following:

- Caching, such as in-network caching, redundancy-aware routing algorithm. These strategies assume that the routers are equipped with data caches. The distribution of the content is necessary to make a decision where to cache which content.
- Content distribution network (CDN): The goal of CDN is a balance between cost and QoS. Its performance heavily depends on the geographical distribution of the contents on the edge servers and the distribution of the user requests. The spatio-temporal traffic generation may help the data management in CDN to decide where to deploy edge servers, the capacity of the edge servers, when to upgrade the server, whether to retire the server, how long different content is to be considered fresh, and so on. In a meanwhile, our traffic generator is also useful for the protocol evaluation that does not require content generation, such as HTTP or TCP. In such cases, the spatio-temporal workload generation can be applied separately.
- Redundancy reduction: Traffic redundancy stems from common end-users activities, such as repeatedly accessing, downloading, distributing and modifying the same or similar information items (documents, data, web and video). Our content generative model is able to evaluate the proposed redundancy reduction algorithms.
- Intrusion detection systems (IDS): Content generation is commonly used for IDS, because it requires deep packet inspection, which compares the headers and payload of network packets against a set of known malicious signatures to determine whether the traffic is benign (the system will ignore) or malicious (the system will alert). Our generator is able to evaluate the IDS under different scenarios by varying the content of the packets (by manipulating the

rule set to generate the content), the distribution of the content, the injection frequency of the malicious traffic, etc.

- Online social network (OSN): As OSNs spring up (e.g., LinkedIn, blog, and Wikipedia), it is important to have a thorough understanding of the user behavior. There is existing work (e.g. [GTC+09]) focusing on the pattern of the user participation and posting behavior, the popularity and quality of the content, and etc. Our traffic generator may reproduce the user behavior and generate various contents for the researchers and industry to better understand the user activities, adjust strategies on how to attract new users and keep existing users, predict the trends of the topics and perform efficient resource.
- Web content generation and delivery: Todays Web sites have developed steadily from document Web (static contents like HTML and images) to application Web and service Web that serve dynamic and complex Web contents and business functions [RaWS09]. Dynamic pages require servers to generate the response content as per the users request before delivering the content back to the user, which introduce extra latency. The traditional content delivery acceleration technologies (e.g., caching and CDN) improved the static Web site performance, however, dynamic Web content brings new challenges. Our traffic generation can facilitate in this area to evaluate the new techniques, such as new caching strategies and delivery approaches to reduce the response time on the server side and the delivery time experienced by the clients.

BIBLIOGRAPHY

[AAB05] Eitan Altman, Konstantin Avrachenkov, and Chadi Barakat. A stochastic model of TCP/IP with stationary random losses. *IEEE/ACM Transactions on Networking (TON)*, 13(2):356–369, April 2005.

[AD96] Jong Suk Ahn and P.B. Danzig. Packet network simulation: speedup and accuracy versus timing granularity. *IEEE/ACM Transactions on Networking (TON)*, 4(5):743–757, October 1996.

[AMS82] D. Anick, D. Mitra, and M. M. Sondhi. Stochastic theory of a data handling system with multiple sources. *The Bell System Technical Journal*, 61(8):1871–1894, October 1982.

[Arc] Internet traffic archive. http://ita.ee.lbl.gov. Last accessed: 2014.

[ASA09] Ashok Anand, Vyas Sekar, and Aditya Akella. Smartre: An architecture for coordinated network-wide redundancy elimination. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, pages 87–98, 2009.

[AV98] P. Abry and D. Veitch. Wavelet analysis of long-range dependent network traffic. *IEEE Transactions on Information Theory*, 44:2–15, 1998.

[BC98] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance. In *Proceedings of the 1998 ACM SIGMETRICS joint International Conference on Measurement and Modeling of Computer Systems*, volume 26, pages 151–160, 1998.

[Ber94] Jan Beran. *Statistics for Long-Memory Processes*. Chapman & Hall, 1994.

[BH02] Francois Baccelli and Dohy Hong. AIMD, fairness and fractal scaling of TCP traffic. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications (INFOCOM '02)*, volume 1, pages 229–238, 2002.

[BH03] Francois Baccelli and Dohy Hong. Flow level simulation of large IP networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM '03)*, volume 3, pages 1911–1921, 2003.

[BKS+10] Vineet Bharti, Pankaj Kankar, Lokesh Setia, Gonca Gursun, Anukool Lakhina, and Mark Crovella. Inferring invisible traffic. In *Proceedings of the 6th International COnference (Co-NEXT '10)*, pages 1–12, 2010.

[Bra89] R. Braden. Requirements for internet hosts – communication layers. *RFC 1122*, 1989.

[BRZL07] Sumitha Bhandarkar, A. L. Narasimha Reddy, Yueping Zhang, and Dimitri Loguinov. Emulating AQM from end hosts. In *Proceedings of Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '07)*, volume 37, pages 349–360, October 2007.

[BTI+02] Chadi Barakat, Patrick Thiran, Gianluca Iannaccone, Christophe Diot, and Philippe Owezarski. A flow-based model for internet backbone traffic. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment (IMW '02)*, pages 35–47, 2002.

[BVG96] J. Bolot and A. Vega-Garcia. Control mechanisms for packet audio in the Internet. *Proceedings of the 15th Annual Joint Conference of the IEEE Computer Societies (INFOCOM '96)*, 1:232–239, 1996.

[CAIa] CAIDA. http://www.caida.org. Last accessed: 2014.

[CAIb] CAIDA Internet Traces. http://www.caida.org/data/passive/. Last accessed: 2014.

[CB95] M.E. Crovella and A. Bestavros. Explaining world wide web traffic self-similarity. Technical Report TR-95-015, Computer Science Department, Boston University, 1995.

[CCG+04] Jin Cao, W. S. Cleveland, Yuan Gao, K. Jeffay, F. D. Smith, and M. Weigle. Stochastic models for generating synthetic HTTP source traffic. *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, 3:1546–1557, 2004.

[CDXL08] Yufeng Chen, Yabo Dong, Zhengtao Xiang, and Dongming Lu. A hybrid simulating framework of TCP traffic at aggregated level. In *3rd International Conference on Communications and Networking in China (ChinaCom '08)*, pages 327–332, 2008.

[CSA00] Neal Cardwell, Stefan Savage, and Thomas Anderson. Modeling TCP latency. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00)*, volume 3, pages 1724–1751, 2000.

[DJ91] P. B. Danzig and S. Jamin. Tcplib: A library of TCP/IP traffic characteristics. Technical Report TR CS-SYS-91-01, USC Networking and Distributed Systemds Laboratory, 1991.

[DPR+08] A. Dainotti, A. Pescapè, P. Salvo Rossi, F. Palmieri, and G. Ventre. Internet traffic modeling by means of hidden markov models. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 52:2645–2662, 2008.

[DPV09] A. Dainotti, A. Pescapè, and G. Ventre. A cascade architecture for DoS attacks detection based on the wavelet transform. *Journal of Computer Security*, 17:945–968, 2009.

[DVJ98] D. Daley and D. Vere-Jones. *An introduction to the thory of point processes*. Springer-Verlag, 1998.

[ECT06] Vijay Erramilli, Mark Crovella, and Nina Taft. An independent-connection model for traffic matrices. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 251–256, 2006.

[ELL09] Miguel Erazo, Yue Li, and Jason Liu. SVEET! A scalable virtualized evaluation environment for TCP. In *Proceedings of the 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities. (TridentCom '09)*, pages 1–10, 2009.

[FGHW99] Anja Feldmann, Anna Gilbert, Polly Huang, and Walter Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication (SIGCOMM '99)*, volume 29, pages 301–313, 1999.

[FJ93] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking (TON)*, 1:397–413, 1993.

[FK03] Sally Floyd and Eddie Kohler. Internet research needs better models. *ACM SIGCOMM Computer Communication Review*, 33(1):29–34, 2003.

[Fla92] P. Flandrin. Wavelet analysis and synthesis of fractional brownian motion. *IEEE Transactions on Information Theory*, 38(2):910–917, 1992.

[FML+03] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S. C. Diot. Packet-level traffic measurements from the sprint ip backbone. *IEEE Network*, 17(6):6–16, 2003.

[FP01] Sally Floyd and Vern Paxson. Difficulties in simulating the Internet. *IEEE/ACM Transactions on Networking (TON)*, 9(4):392–403, 2001.

[FPP+03] R. M. Fujimoto, K. Perumalla, A. Park, H. Wu, M. H. Ammar, and G. F. Riley. Large-scale network simulation: How big? How fast? In *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS '03)*, pages 116–123, 2003.

[FTT02] Kensuke Fukuda, Hideki Takayasu, and Misako Takayasu. Spatial and temporal behavior of congestion in Internet traffic. *Fractals*, 7:147–163, 2002.

[FW02] Qiang Fu and L. White. The impact of background traffic on TCP performance over indirect and direct routing. In *The 8th International Conference on Communication Systems (ICCS '02)*, volume 1, pages 594–598, 2002.

[GGT00] Y. Guo, W. Gong, and D. Towsley. Time-stepped hybrid simulation (TSHS) for large scale networks. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00)*, volume 2, pages 441–450, 2000.

[GLT04] Y. Gu, Y. Liu, and D. Towsley. On integrating fluid models with packet simulation. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, volume 4, pages 2856–2866, 2004.

[GTC+09] Lei Guo, Enhua Tan, Songqing Chen, Xiaodong Zhang, and Yihong (Eric) Zhao. Analyzing patterns of user content generation in online social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09)*, pages 369–378, 2009.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11:10–18, June 2009.

[HGAS13] Dongsu Han, Robert Grandl, Aditya Akella, and Srinivasan Seshan. FCP: A flexible transport framework for accommodating diversity. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 135–146, 2013.

[HKS99] H. Hlavacs, G. Kotsis, and C. Steinkellner. Traffic source modeling. Technical Report TR-99101, Institute of Applied computer Science and Information Systems, University of Vienna, 1999.

[HLRX07] Sangtae Ha, Long Le, Injong Rhee, and Lisong Xu. Impact of background traffic on performance of high-speed TCP variant protocols. *Computer Networks*, 51(7):1748–1762, 2007.

[KCR08] Changhoon Kim, Mattew Caesar, and Jennifer Rexford. Floodless in SEATTLE: A scalable ethernet architecture for large enterprises. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 3–14, 2008.

[KMM+11] Nupur Kothari, Ratul Mahajan, Todd Millstein, Ramesh Govindan, and Madanlal Musuvathi. Finding protocol manipulation attacks. In *Proceedings of the ACM SIGCOMM 2011 conference*, pages 26–37, 2011.

[KPF05] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. BLINC: Multilevel traffic classification in the dark. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '05)*, pages 229–240, 2005.

[KSCK96] G. Kesidis, A. Singh, D. Cheung, and W.W. Kwok. Feasibility of fluid-driven simulation for ATM network. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '96)*, volume 3, pages 2013–2017, 1996.

[KSWU03] Cameron Kiddle, Rob Simmonds, Carey Williamson, and Brian Unger. Hybrid packet/fluid flow network simulation. In *Proceedings of the 17th Workshop on Parallel and Distributed Simulation (PADS'03)*, pages 143–152, 2003.

[KT04] Inas Khalifa and Ljiljana Trajkovic. An overview and comparison of analytical TCP models. In *Proceedings of the 2004 International Symposium on Circuits and Systems (ISCAS '04).*, volume 5, pages 469–472, 2004.

[LDK10] Myungjin Lee, Nick Duffield, and Ramana Rao Kompella. Not all microseconds are equal: fine-grained per-flow measurements with reference latency interpolation. In *Proceedings of the ACM SIGCOMM 2010 conference*, pages 27–38, 2010.

[LFG+01] B Liu, D. R. Figueiredo, Y. Guo, J. kurose, and D. Towsley. A study of networks simulation efficiency: fluid simulation vs. packet-level simulation. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '01)*, pages 1244–1253, 2001.

[Liu06] Jason Liu. Packet-level integration of fluid TCP models in real-time network simulation. In *Proceedings of the 38th Winter Simulation Conference (WSC '06)*, pages 2162–2169, 2006.

[LM05] Song Luo and Gerald A. Marin. Realistic internet traffic simulation through mixture modeling and a case study. In *Proceedings of the 37th Winter Simulation Conference (WSC '05)*, pages 2408–2416, 2005.

[LPC+04] Anukool Lakhina, Konstantina Papagiannaki, Mark Crovella, Christophe Diot, and Kolaczyk. Structural analysis of network traffic flows. In *Proceedings of the 2004 SIGMETRICS joint International Conference on Measurement and Modeling of Computer Systems*, pages 61–72, 2004.

[LPM+03] Yong Liu, Francesco Lo Presti, Vishal Misra, Don Towsley, and Yu Gu. Fluid models and solutions for large-scale IP networks. In *Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 91–101. ACM, 2003.

[LTWW94] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[LYPN02] M. Liljenstam, Y. Yuan, B. J. Premore, and D. Nicol. A mixed abstraction level simulation model of large-scale Internet worm infestations. In *Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS '02)*, pages 109–116, 2002.

[LYX10] Xin Liu, Xiaowei Yang, and Yong Xia. NetFence: preventing Internet denial of service from inside out. In *Proceedings of the ACM SIGCOMM 2010 conference*, pages 255–266, 2010.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[MAW] MAWI Traffic Archive. http://tracer.csl.sony.co.jp/mawi/. Last accessed: 2014.

[MGG+05] Marco Ajmone Marsan, Michele Garetto, Paolo Giaccone, Emilio Leonardi, Enrico Schiattarella, and Alessandro Tarello. Using partial differential equations to model TCP mice and elephants in large IP networks. *IEEE/ACM Transactions on Networking (TON)*, 13(6):1289–1301, 2005.

[MGT00] Vishal Misra, Wei-Bo Gong, and Don Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '00)*, pages 151–160, 2000.

[MGT03] Vishal Misra, Wei-Bo Gong, and Don Towsley. Fluid models and solutions for large-scale IP networks. In *Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and Modeling of Computer Systems (SIGMETRICS '03)*, pages 91–101, 2003.

[MHL+04] Anthony Mcgregor, Mark Hall, Perry Lorier, James Brunskill Anthony Mcgregor, Mark Hall, Perry Lorier, and James Brunskill. Flow clustering using machine learning techniques. In *Proceedings of the 5th International Workshop on Passive and Active Network Measurement (PAM '04)*, pages 205–214, 2004.

[MLMB01] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. BRITE: An approach to universal topology generation. In *Proceedings of the 9th International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '01)*, page 346, 2001.

[MN68] Benoit B. Mandelbrot and John W. Van Ness. Fractional brownian motions, fractional noises and applications. *SIAM Review*, 10(4):422–437, 1968.

[MSMO97] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM Computer Communication Review*, 27(3):67–82, 1997.

[MSZ02] Marco Mellia, Ion Stoica, and Hui Zhang. TCP model for short lived flows. *IEEE Communications Letters*, 6:85–87, 2002.

[MTS+02] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '02)*, pages 161–174, 2002.

[Nic01] David M. Nicol. Fluid simulation: Discrete event fluid modeling of TCP. In *Proceedings of the 33rd Winter Simulation Conference (WSC '01)*, pages 1291–1299, 2001.

[Nic02] David Nicol. DARPA NMS baseline network topology. http://www.ssfnet.org/Exchange/gallery/baseline/index.html, 2002. Last accessed: 2014.

[NLA] Nlanr. http://www.nlanr.net. Last accessed: 2014.

[NST05] Antonio Nucci, Ashwin Sridharan, and Nina Taft. The problem of synthetically generating IP traffic matrices: initial recommendations. *ACM SIGCOMM Computer Communication Review*, 35:19–32, 7 2005.

[NY04] David M. Nicol and Guanhua Yan. Discrete event fluid modeling of background TCP traffic. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 14(3):211–250, 2004.

[Pax94] Vern Paxon. Empirically derived analytic models of wide-area TCP connections. *IEEE/ACM Transactions on Networking (TON)*, 5(4):316–336, 1994.

[Pax97a] Vern Paxson. End-to-end Internet packet dynamics. *ACM SIGCOMM Computer Communication Review*, 27(4):139–152, 1997.

[Pax97b] Vern Paxson. Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic. *ACM SIGCOMM Computer Communication Review*, 27:5–18, 1997.

[PF94] Vern Paxon and Sally Floyd. Wide area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking (TON)*, 3(3):226–244, 1994.

[PFTK00] Jitendra Padhye, Victor Firoiu, Donald F. Towsley, and James F. Kurose. Modeling TCP Reno performance: A simple model and its empirical validation. *IEEE/ACM Transactions on Networking (TON)*, 8:133–145, 2000.

[PG08] Maxim Podlesny and Sergey Gorinsky. RD network services: differentiation through performance incentives. In *Proceedings of the ACM SIGCOMM 2008 conference on Data Communication (SIGCOMM '08)*, pages 255–266, 2008.

[PKV07] Peter Pocta, Peter Kortis, and Martin Vaculik. Impact of background traffic on speech quality in VoWLAN. *Advances in Multimedia*, 2007(1), 2007.

[PMH09] Pavlos Papageorge, Justin Mccann, and Michael Hicks. Passive aggressive measurement with MGRP. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data communication (SIGCOMM '09)*, pages 255–266, 2009.

[PRI11] PRIME Research Group. Prime. http://www.primessf.net/, 2011.

[PTZD03] Konstantina Papagiannaki, Nina Taft, Zhi-Li Zhang, and Christophe Diot. Long-term forecasting of internet backbone traffic: Observations and initial models. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM '03)*, pages 1178–1188, 2003.

[RaWS09] Jayashree Ravi and Zhifeng Yu adn Weisong Shi. A survey on dynamic web content generation and delivery techniques. *Journal of Network and Computer Applications*, 32(5):943–960, 2009.

[RFI02] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the Gnutella network: properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Special Issue on Peer-to-Peer Networking*, 6(1):50–57, 2002.

[RGK+02] Matthew Roughan, Albert Greenberg, Charles Kalmanek, Michael Rumsewicz, Jennifer Yates, and Yin Zhang. Experience in measuring backbone traffic variability: models, metrics, measurements and meaning. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment (IMW '02)*, pages 91–92, 2002.

[RJF02] George F. Riley, Talal M. Jaafar, and Richard Fujimoto. Integrated fluid and packet network simulations. In *Proceedings of the IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer*

Telecommunication Systems (MASCOTS '02), pages 511–518. IEEE Computer Society, 2002.

[RSSD04] Matthew Roughan, Subhabrata Sen, Oliver Spatscheck, and Nick Duffield. Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement (IMC'04)*, pages 135–148, 2004.

[SB04] Joel Sommers and Paul Barford. Self-configuring network traffic generation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet Measurement (IMC '04)*, pages 68–81, 2004.

[SBE+11] Joel Sommers, Rhys Alistair Bowden, Brian Eriksson, Paul Barford, Matthew Roughan, and Nick G. Duffield. Efficient network-wide flow record generation. In *Proceedings of the 30th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '11)*, pages 2363–2371, 2011.

[SCK03] W. Shi, E. Collins, and V. Karamcheti. Modeling object characteristics of dynamic Web content. *Journal of Parallel and Distributed Computing-Scalable Web Services and Architecture*, 63:963–980, 10 2003.

[SCK+07] Jinsheng Sun, Sammy Chan, King-Tim Ko, Guanrong Chen, and Moshe Zukerman. Instability effects of two-way traffic in a TCP/AQM system. *Computer Communications*, 30(10):2172–2179, 2007.

[SKV04] B. Sikdar, S. Kalyanaraman, and K. S. Vastola. Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno and SACK. *IEEE/ACM Transactions on Networking (TON)*, 11(6):959–971, 2004.

[SLN94] M. Stoksik, R. Lane, and D. Nguyen. Accurate synthesis of fractional brownian motion using wavelets. *Electronics Letters*, 30(2):383–384, 1994.

[SMWA04] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions on Networking (TON)*, 12(1):2–16, 2004.

[SYB04] Joel Sommers, Vinod Yegneswaran, and Paul Barford. A framework for malicious workload generation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement (IMC '04)*, pages 82–87, 2004.

[TMW97] Kevin Thompson, Gregory J. Miller, and Rick Wilder. Wide-area Internet traffic patterns and characteristics. *IEEE Network*, 11:10–23, 1997.

[TR13] Paul Tune and Matthew Roughan. Internet traffic matrices: A primer. *Recent Advances in Networking*, 1, 2013.

[VK11] V. Valgenti and M. S. Kim. Simulating content in traffic for benchmarking intrusion detection systems. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques (SIMUTools '11)*, pages 44–50, 2011.

[VK12] Victor C. Valgenti and Min Sik Kim. An application-level content generative model for network applications. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques (SIMUTools '12)*, pages 47–56, 2012.

[VV06] Kashi Venkatesh Vishwanath and Amin Vahdat. Realistic and responsive network traffic generation. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '06)*, pages 111–122, 2006.

[VV08] Kashi Venkatesh Vishwanathvishwanath and Amin Vahdat. Evaluating distributed systems: Does background traffic matter? In *Proceedings of 2008 Annual Technical Conference (ATC '08)*, pages 227–240. USENIX Association, 2008.

[VYW+02] Amin Vahdat, Ken Yocum, Kevin Walsh, Priya Mahadevan, Dejan Kostic, Jeff Chase, and David Becker. Scalability and accuracy in a large scale network emulator. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, volume 36, pages 271–284, 2002.

[WAF99] Mengzhi Wang, Anastassia Ailamaki, and Christos Faloutsos. Capturing the spatio-temporal behavior of real traffic data. *Performance Evaluation*, 49:23–31, 1999.

[WAHC+06] Michele C. Weigle, Prashanth Adurthi, Félix Hernández-Campos, Kevin Jeffay, and F. Donelson Smith. Tmix: A tool for generating realistic TCP application workloads in NS-2. *ACM SIGCOMM Computer Communication Review*, 36(3):67–76, 2006.

[WMK06] Songjie Wei, Jelena Mirkovic, and Ezra Kissel. Profiling and clustering Internet hosts. In *Proceedings of the 2006 International Conference on Data Mining (DMIN '06)*, pages 269–275, 2006.

[WWM+10] Ye Wang, Hao Wang, Ajay Mahimkar, Richard Alimi, Yin Zhang, Lili Qiu, and Yang Richard Yang. R3: resilient routing reconfiguration. In *Proceedings of the ACM SIGCOMM 2010 conference*, pages 291–302, 2010.

[XlZB05] Kuai Xu, Zhi li Zhang, and Supratik Bhattacharyya. Profiling Internet backbone traffic: Behavior models and applications. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '05)*, pages 169–180, 2005.

[YM05] Jian Yuan and Kevin Mills. A cross-correlation based method for spatial-temporal traffic analysis. *Performance Evaluation*, 61:163–180, 2005.

[YMKT99] M Yajnik, Sue Moon, J Kurose, and D Towsley. Measurement and modelling of the temporal dependence in packet loss. In *Proceedings of 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '99)*, volume 1, pages 345–352, 1999.

[ZGGR05] Yin Zhang, Zihui Ge, Albert Greenberg, and Matthew Roughan. Network anomography. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement (IMC '05)*, pages 317–330, 2005.

[ZNA03] Moshe Zukerman, Timothy D. Neame, and Ronald G. Addie. Internet traffic modeling and future technology implications. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM '03)*, volume 1, pages 587–596, 2003.

[ZRDG03] Yin Zhang, Matthew Roughan, Nick Duffield, and Albert Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems (SIGMETRICS '03)*, pages 206–217, 2003.

[ZRLD03] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '03)*, pages 301–312, 2003.

[ZRLD05] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. Estimating point-to-point and point-to-multipoint traffic matrix: An information-theoretic approach. *IEEE/ACM Transactions on Networking (TON)*, 13:947–960, 2005.

[ZRWQ09] Y. Zhang, M. Roughan, W. Willinger, and L. Qui. Spatio-temporal compressive sensing and internet traffic matrices. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 267–278, 2009.

VITA

TING LI

March 11, 1982	Born, Tianjin, China
2000–2004	B.S., Communication Engineering Nankai University Tianjin, China
2004–2007	M.S., Communication and Information System Nankai University Tianjin, China
2008–2014	Doctoral Candidate Florida International University Miami, Florida

PUBLICATIONS

T. Li, and J. Liu. Cluster-Based Spatio-Temporal Background Traffic Generation for Network Simulation. Submitted to ACM Transactions on Modeling and Computer Simulation (TOMACS).

J. Liu, Y. Liu, Z. Du, T. Li and J. Wang. GPU-Assisted Hybrid Network Traffic Model. ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (PADS'14).

T. Li, N. Van Vorst and J. Liu. A Fast Rate-Based TCP Traffic Model. SIMULATION: Transactions of the Society for Modeling and Simulation International, Volume 89, Issue 4, Pages 466-480, April 2013.

M. Erazo, T. Li, J. Liu and S. Eidenbenz. Toward Comprehensive and Accurate Simulation Performance Prediction of Parallel File Systems. The 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Pages 1-12, Boston, Massachusetts, June 25-28, 2012.

T. Li, N. Van Vorst, R. Rong, and J. Liu. Simulation Studies of OpenFlow-Based In-Network Caching Strategies. Proceedings of the 15th Communications and Networking Simulation Symposium (CNS), Pages 50-55, Orlando, FL, March 26-29, 2012. The best paper award.

N. Van Vorst, T. Li, and J. Liu, How Low Can You Go? Spherical Routing for Scalable Network Simulations. The 19th Annual Meeting of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), Pages 259-268, Singapore, July 25-27, 2011.

T. Li and J. Liu. A Fluid Background Traffic Model. IEEE International Conference on Communications (ICC), Pages 1-6, Dresden, Germany, June 14-18, 2009.