

# A Methodology for the Optimal Configuration of TCP Traffic in Network Simulations under Link Load Constraints \*

Qi He, Constantinos Dovrolis, Mostafa Ammar  
*qhe, dovrolis, ammar@cc.gatech.edu*  
College of Computing, Georgia Tech

## Abstract

*Given that over 90% of the Internet load is carried by TCP, most network simulation studies use TCP flows to generate the background traffic. A basic, but unresolved, question however is: how can one decide how many TCP flows to simulate from one network node to another? Simulating too many flows on a link can cause an unrealistically high loss rate on that link, while simulating too few flows can result in undesirably light load conditions. Similarly, to simulate realistic network conditions, one has to carefully control the load distribution on various network links (e.g., edge vs. core links), as well as the hop count (path length) of the simulated TCP flows. Previous simulation studies have dealt with these issues in a trial-and-error manner, experimenting with several traffic configurations until a realistic distribution of link load and loss rate is achieved. In this paper, we present a methodology that determines the number of TCP flows that should be simulated between each pair of nodes in a network, based on the network topology, a specification of the utilization and loss rate for certain links, and an average number of hops for the TCP flows. Our methodology is based on a linear program formulation that, while meeting the utilization and loss rate specifications, minimizes the number of required TCP flows. This optimization criterion minimizes the memory requirement of the simulation. Our evaluations show that the proposed methodology can closely approximate the specified link conditions in terms of utilization and loss rate. We also analyze the largest approximation errors and reveal their causes.*

## 1 Introduction

Background traffic is a key component of network simulations, as it shapes the network environment under which network mechanisms or protocols are evaluated. As such, network traffic modeling and generation have attracted

much attention in the literature [1, 4, 12, 17].

To gain meaningful insights from network simulations, it is important to have realistic and controlled network conditions [10], in terms of link utilization, queuing delay and loss rate. The ability to explicitly control link conditions is useful for studying a subject under a spectrum of controlled path conditions. For example, it is demonstrated that link utilization has a significant impact on the performance of AQM schemes [13]. Explicit control of link conditions also enables the simulation of important network characteristics such as diversity, locality and correlation of path conditions, which can affect applications such as overlay routing [2], Content Distribution Networks, and reliable multicast [8]. For instance, it is shown in [2] that the locality of congestion and path conditions in the Internet influences the relative performance of alternative overlay routing optimizations.

Network conditions are complex functions of both the offered traffic and the physical configuration of the network, namely the topology, link capacities and router buffer sizes. Whereas models for both exist [4, 5] and are used for simulation, they may lead to unrealistic link conditions if the offered load is not tailored to the network configuration, as noted by some previous work [17]. Unfortunately, as we will discuss shortly, existing traffic generation methods are in general isolated from the network configuration and lack considerations for the link conditions that ensue.

In this work, we are concerned with generating traffic for network simulations such that the resulting conditions on certain links, which we call *target links*, are as specified by the simulation users. We propose a methodology for the optimal configuration of traffic under the specified link load constraints. The algorithm takes as input a detailed network configuration and the target link conditions. It then configures the traffic intensity between each pair of source and destination nodes, such that all target link conditions are achieved at the same time.

There are traditionally two categories of traffic generation: *open-loop* [14] and *closed-loop* [1, 4, 6, 17]. With both categories of traffic generation methods, the focus typically has been on the accurate modeling either of the

---

\*This paper has been supported in part by the NSF under grant numbers ANI-9973115, ANI-9977544 and ANI-0136939. All opinions, findings and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

packet arrival process or of the traffic source behavior, so as to capture network traffic characteristics such as long range dependence and self-similarity [9]. None of the existing schemes sufficiently addresses first-order link characteristics such as average utilization and loss rate, which as we have mentioned are quite relevant to network studies. An oft-adopted traffic configuration for large-scale wide-area network simulation, for example, is to randomly select source/destination pairs, and add traffic between each pair based on a traffic source model [17], possibly with an empirical average traffic rate. For a network where links have limited and diverse capacities, such a configuration could result in unpredictable and unrealistic link conditions, e.g., unusually low utilization or extremely high congestion.

The need to explicitly control link conditions has been occasionally acknowledged by previous work. For example, in order to control the utilization of a simulated link, it has been suggested that one can tune the traffic iteratively until the desired utilization is reached [17]. Such an approach is expensive because of the iterative simulations, and it works only for simple topologies. As to queuing delay and loss rate, some network simulation and emulation tools provide a stock of stochastic models for users to control them [3]. Since losses and queuing delays generated by these models do not result from a TCP workload, however, this approach defeats the goal of closed-loop simulation.

The traffic generation methodology we propose in this work is for closed-loop network simulations. In particular, it uses a mixture of persistent TCP flows and ON/OFF UDP flows, which is the traffic model many network studies use [7, 8]. The UDP flows generate a small fraction of the traffic (typically less than 10%) and they are intended to simulate applications such as DNS and streaming audio/video. Our goal is then to configure the number of TCP flows between each pair of nodes, such that the user-specified conditions on all target links are attained. We model the traffic generation problem as a linear program, where each variable represents the number of TCP flows from a particular source to a particular destination, with constraints on the target link conditions. While obtaining the target link conditions, the algorithm generates traffic in such a way that the average flow hop count is as specified, and the memory requirement of the simulation, determined mostly by the number of TCP flows, is minimized. Our algorithm exploits two basic facts: (1) the throughput of a TCP flow can be derived based on the load and loss rate of the links on its path [15]; (2) the load on a link, in turn, is the sum of the throughput of all flows traversing it.

The problem stated above is related to the traffic matrix estimation algorithm in network research [18]. The challenge, objectives and context are nonetheless very different. Our methodology can also be conceived as inferring TCP flow configurations based on the specified link condi-

tions. In that respect, it is also related to the work reported in RAMP [12]. That work recovers path and flow information from packet traces and obtains a full traffic profile including the traffic mix, flow size distribution and arrival rate, which can then be used to populate network simulations. Clearly, the goal of RAMP is not to configure traffic to match user-specified link conditions, far less to do so for a network of links.

The contributions of this paper can be summarized as follows:

- We develop a model to configure traffic for closed-loop network simulations such that specific link conditions are attained on certain links. In doing that, we also maintain an average hop count in the generated traffic, and minimize the number of required TCP flows so as to reduce the memory requirement of the simulation.
- We evaluate and analyze the accuracy of the algorithm and provide insight on the causes of the major errors.

The rest of the paper is organized as follows. In Section 2, we specify the traffic generation problem we address in this paper. Section 3 presents a linear program formulation of the problem. Section 4 evaluates the accuracy of the proposed traffic generation scheme. We discuss future work and conclude the paper in Section 5.

## 2 Problem Statement

In this section, we describe in detail the traffic generation problem that we consider in this work.

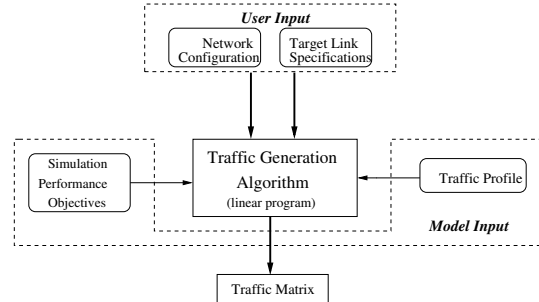


Figure 1. Traffic Generation Context

### 2.1 Overview of the Traffic Generation Context

To formally define the traffic generation problem, we need to precisely state its context with all the relevant components. The context we propose in this work is outlined

in Figure 1. At the heart of this context is a traffic generation algorithm that we will elaborate on in Section 3. We distinguish two types of inputs to this algorithm:

(1) *User inputs* are parameters that concern the simulation user and, in general, they are independent of the traffic generation process. One of the user inputs is the network configuration, and the other is a list of target links in the network with specified link conditions.

(2) *Model inputs* are parameters that we add to influence certain aspects of the traffic generation process. First, a traffic profile is introduced to add some realism in the generated traffic, and it could reflect characteristics such as the mixture of different types of traffic. Second, we specify a particular aspect of the simulation performance to optimize. If simulation speed is the primary concern, e.g., we can minimize the total packet hop count, hence the number of events per simulated unit of time.

Collectively, the model inputs and the traffic generation algorithm comprise our traffic generation scheme. The output of the traffic generation algorithm specifies how many TCP flows to configure between each pair of nodes so as to obtain the user specified link conditions.

## 2.2 Traffic Generation Context Specification

The specific traffic generation context we use in this work can be described as follows:

- **Input 1: Network configuration.** The following parameters of a network are provided as inputs:

(a) Topology and routing. The network consists of  $V$  nodes and  $N$  links. Network routing is fully specified by the following function ( $1 \leq i, j \leq V, 1 \leq k \leq N$ ):

$$R_{ij}(k) = \begin{cases} 1 & \text{if link } k \text{ is in the path from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases}$$

(b) Detailed configuration of the links. Each link  $L_k$  ( $1 \leq k \leq N$ ) is described by a 5-tuple  $(S_k, D_k, C_k, B_k, T_k)$ , representing source node, destination node, link capacity, router buffer size (in packets) and propagation delay, respectively. Notice that the links are *directional*.

- **Input 2: Target link specifications.** This input specifies a list of  $N_g$  ( $\leq N$ ) target links. The target link conditions we consider include the utilization, and *average* loss rate.

Each target link has two additional characteristics: target utilization  $\rho$  (in  $[0, 1]$ ) and target loss rate  $p$  (in  $[0, 1]$ ). We assume that  $\rho < 1$  prescribes  $p = 0$ , and that  $p > 0$  prescribes  $\rho = 1$ , i.e., lossy links are fully utilized and underutilized links are lossless. These are

legitimate assumptions when the router buffers are sufficiently configured [7]<sup>1</sup>. There are no specific link conditions for the rest of the links, which we call *non-target links*, but we make it an implicit specification that the load on each of them is lower than its capacity. This prevents non-target links from limiting the traffic passing through target links.

- **Input 3: Traffic profile.** The traffic profile characterizes the traffic from the following aspects:

(a) Traffic mix. The traffic on any link consists of a number of persistent TCP flows and Pareto ON/OFF UDP streams. UDP traffic accounts for a small fraction  $u$  (typically less than 10%) of the total load on every link. In addition, the presence of UDP traffic eliminates the synchronization among persistent TCP flows, usually an artifact of TCP simulations [10].

(b) TCP flow characteristics. A flow from node  $i$  to  $j$  is characterized by the 2-tuple  $(H_{ij}, W_{ij})$ , where  $H_{ij}$  is the path hop count determined by routing  $\{R_{ij}(k)\}$ .  $W_{ij}$  is the maximum TCP congestion window configured on a per source/destination pair basis. When the path from  $i$  to  $j$  is lossless,  $W_{ij}$  follows a given distribution; when the path is lossy,  $W_{ij}$  is large enough for the flow to be congestion-limited on it. This latter assumption about the configuration needs more explanations, which we defer to the end of Section 3.3. Finally, we assume that TCP data segments and UDP packets of all flows have the same size  $S$ .

(c) Traffic between any pair of nodes is allowed, subject to the constraint that the average hop count among all TCP flows is  $H_t$ .

- **Input 4: Simulation performance objective.** In this work, we minimize the simulation memory requirement. This is motivated by the fact that the large memory requirement often preempts the execution of even medium-scale network simulations.
- **Output: a traffic matrix**  $\{A\}_{V \times V}$  where  $A_{ij}$  represents the number of TCP flows from node  $i$  to  $j$ . The user will be able to attain the link conditions he/she specifies for the target links, as long as he/she configures  $A_{ij}$  TCP flows from node  $i$  to node  $j$ , for all pairs of  $i$  and  $j$ .

**Problem Statement.** With all the components defined above, the traffic generation problem we aim to solve can be stated as follows:

<sup>1</sup>A common rule is to configure a router buffer at least twice as large as the bandwidth delay product of the flows with the largest RTT.

Given a network with  $V$  nodes,  $N$  edges and specified routing  $\{R_{ij}(k)\}$ , and a set of  $N_g$  target links with specified link utilization and loss rate, generate a matrix of TCP flows  $\{A\}_{V \times V}$  such that 1) the link conditions on all target links are obtained, 2) the average hop count across all flows is as specified by  $H_t$ , and that 3) the simulation memory requirement is minimized.

For quick reference, Table 1 lists some notations that we use in the rest of the paper.

$V$ : number of nodes	$N$ : number of links
$C$ : link capacity	$T$ : propagation delay of a link
$B$ : router buffer (in packets)	$\hat{T}$ : RTT of a flow
$p$ : loss rate of a target link	$T_q$ : queuing delay on a lossy link
$S$ : TCP data segment size	$\rho$ : utilization of a target link
$H_t$ : average flow hop count	$b$ : segments per ACK in TCP
$R_{ij}(k)$ : whether link $k$ is in the path from $i$ to $j$	
$U_{ij}$ : throughput of a TCP flow from $i$ to $j$	
$A_{ij}$ : number of TCP flows from $i$ to $j$ to be solved for	
$H_{ij}$ : hop count of the path from $i$ to $j$	
$u$ : fraction of UDP traffic on each link	
$W$ : TCP maximum congestion window (in packets)	

**Table 1. Terms and Notations**

### 3 Traffic Generation Algorithm

In this section, we develop algorithms to solve the traffic generation problem stated in Section 2.

#### 3.1 Single Link Example

The traffic generation algorithm we propose in this work is based on two insights: (i) the throughput of a TCP flow can be derived based on the load conditions of its path, in addition to some characteristics, such as the maximum congestion window, of the flow itself [15]; (ii) the load of a link, in turn, is the sum of the throughput of all flows traversing the link. We call the second property *traffic balance*. In this subsection, we use a single-link example to demonstrate how we can use these insights to establish the relation between the utilization/loss rate of a link and the number of TCP flows traversing it.

The sample target link is  $(S_0, D_0, C, B, T, \rho, p)$ . All flows traversing it are from  $S_0$  to  $D_0$  and have the same maximum congestion window  $W$ . The traffic generation problem in this context is simply to obtain the number of flows needed to reach the target  $\rho$  or  $p$ .

**TCP Throughput Model:** When the path taken by a flow is lossless, the throughput  $U$  of the flow is limited by its maximum congestion window  $W$  as in:  $U = \frac{WS}{\hat{T}}$ , where  $S$  is the TCP segment size and  $\hat{T}$  is the RTT of the flow.

For lossy paths, we use the model by Padhye et al. [15], which describes TCP throughput as:

$$U = \min\left\{\frac{S}{\hat{T}\sqrt{\frac{2bp}{3}} + T_0 \min(1, \sqrt{\frac{3bp}{8}})p(1 + 32p^2)}, \frac{WS}{\hat{T}}\right\}, \quad (1)$$

where  $T_0$  is the TCP retransmission timeout period and  $b$  is the number of new segments released per new ACK.

**Target Utilization:** We first consider the flow configuration problem when the target link is lossless and has a target utilization  $\rho < 1$ . In this case, the TCP flows that constitute its load are window-limited. Notice that the RTT of a flow in this uncongested link case is simply twice the one way link propagation delay  $T$ . Therefore, the throughput  $U$  of each flow can be derived as:  $U = \frac{WS}{2T}$ .

The number of TCP flows,  $A$ , that are needed to reach the target utilization  $\rho$  should satisfy traffic balance (2):

$$AU = A \frac{WS}{2T} = C\rho \quad (2)$$

We can then solve (2) for  $A$ .

**Target Loss Rate:** If the target link is lossy, the throughput of a flow traversing it is expressed by (1). The RTT in this case, includes the queuing delay  $T_q$  on the congested link, which can be approximated by  $\frac{BS}{C}$ , because congestion-limited TCP flows tend to fill the buffer of their bottleneck links [7]. The RTT  $\hat{T}$  is then modeled as:

$$\hat{T} = 2T + T_q = 2T + \frac{BS}{C} \quad (3)$$

As long as the buffer is sufficiently provisioned, a lossy link can be assumed to be fully utilized [7]. Therefore, the traffic balance equation for a lossy link is as follows:

$$AU = C \quad (4)$$

From (1), (3) and (4), we can solve for the number of flows  $A$  to obtain the target loss rate  $p$ .

#### 3.2 Problem Formulation

Based on the above discussions, we now formulate the problem stated in Section 2 as a linear program consisting of the following elements:

**Traffic Balance Constraints:** A traffic balance constraint is needed for each link to relate its load to the throughput of the flows traversing it. The traffic constraints for different links comprise a system of  $N_g$  equations (each corresponding to a target link) and  $(N - N_g)$  inequalities (each

corresponding to a non-target link) in the following form ( $1 \leq k \leq N, 1 \leq i, j \leq V$ ):

$$\begin{cases} \sum_{i,j} A_{ij} U_{ij} R_{ij}(k) = (1-u) \rho \cdot C_k & \text{if } k \text{ is a target link} \\ \sum_{i,j} A_{ij} U_{ij} R_{ij}(k) < (1-u) \cdot C_k & \text{if } k \text{ is a non-target link} \\ A_{ij|i=j} = 0 \\ A_{ij|i \neq j} \geq 0 \end{cases} \quad (5)$$

Note that  $U_{ij}$  is the throughput of a TCP flow from node  $i$  to  $j$ . In principle, the variables  $A_{ij}$ , representing numbers of TCP flows, should be integers. Modeling the problem as an integer program, however, could make the problem computationally expensive or infeasible to solve for large-scale networks. Therefore we use a linear program as an approximation. We will show in Section 4 that the resulting errors are noticeable only to certain links and that they are mostly acceptable.

#### Simulation Performance Optimization Objective:

With a given network topology, the memory cost of a simulation using tools such as *ns2* [3] depends mostly on the number TCP flow objects. Therefore, our objective to minimize simulation memory requirement is translated into minimizing the total number of TCP flows. With the addition of the following optimization objective:

$$\text{Minimize} \quad \sum_{i,j} A_{ij}, \quad (6)$$

we now have a linear optimization program that could render a unique solution:

#### Additional Constraint on the Average Flow Hop Count:

We also add an average flow hop count constraint to the existing linear program (see explanations in Section 2.2). This is expressed as follows:

$$\frac{\sum_{i,j} A_{ij} H_{ij}}{\sum_{i,j} A_{ij}} = H_t, \quad \text{i.e.,} \quad \sum_{i,j} A_{ij} (H_{ij} - H_t) = 0 \quad (7)$$

Along with (5) and (6), Equation (7) defines a linear program, (5) and (7) as the constraints and (6) as the objective, for the traffic generation problem described in Section 2.

### 3.3 Populating the Linear Program

In this subsection, we discuss some issues related to obtaining accurate estimates of the required parameters, mainly  $\{U_{ij}\}$ , to populate the linear program defined above.

**Throughput Matrix  $\{U_{ij}\}$ :** To calculate  $U_{ij}$ , we need to have accurate estimates of the loss rate and RTT experi-

enced by a flow from  $i$  to  $j$ . This, however, is not as straightforward as in the single link configuration.

- **RTT.** The RTT of a TCP flow should include the queuing delays both in the forward direction and in the reverse direction (incurred by ACKs). Consider a congested link  $L$  ( $S_0, D_0, C, B, T, \rho_f=1, p>0$ ), and suppose the utilization in its reverse direction is  $\rho_r$ . We found that  $T_q$  can be modeled as follows:

$$T_q = \frac{(r(S + S_{hdr}) + (1-r)S_{ack}) \cdot B \cdot \theta}{C} \quad (8)$$

$$r = \frac{\rho_f}{\rho_f + \frac{\rho_r}{b}} \quad (9)$$

where  $S_{hdr}$  and  $S_{ack}$  are the sizes of the TCP/IP header and of the TCP ACKs, respectively. The reason to use (8) instead of  $T_q = \frac{BS}{C}$  is that there are a significant fraction of packets in the queue of  $L$  that do not have a size of  $S$  and thus do not contribute a queuing delay of  $\frac{S}{C}$ . Those are the ACKs for the TCP data traffic in the reverse direction. Given that every  $b$  data packets result in one ACK, and that the ratio of TCP packet count on  $L$  vs. that on its reverse link is approximated by  $\frac{\rho_f}{\rho_r}$ ,  $r$  in (9) approximates the fraction of TCP data packets among all packets on  $L$ . Therefore,  $r(S + S_{hdr}) + (1-r)S_{ack}$  approximates the average packet size on  $L$ . The term  $\theta$  in (8) accounts for the fact that the queue is not always full. We use  $\theta = 0.8$  in the rest of the paper, while any other value between 0.75 and 1 works similarly well. For the model with a fraction  $u$  of UDP traffic, we can simply replace (9) by the following:

$$r = \frac{\rho_f}{\rho_f + \frac{\rho_r(1-u)}{b}},$$

assuming that the UDP packet size is also  $S$ .

- **Loss rate.** Since TCP is much less sensitive to ACK losses than to data packet losses [11], we only consider the data packet losses in the forward direction. We use an optimistic calculation of the loss rate when multiple bottlenecks exist on its path [16], i.e., we take the largest loss rate among the bottlenecks as the loss rate for a flow.

**Average Hop Count  $H_t$ :** Occasionally, we need to adjust the parameters in the traffic profile based on the user inputs. One such parameter is the target average hop count  $H_t$ , for which an improper value could make the linear program in Section 3.2 infeasible. For instance, if  $H_t$  is larger than the length of the longest path in the network, there is no way we can configure the flows such that the average flow hop count is  $H_t$ .

**Maximum Congestion Window for Flows on Lossy Paths:** The second such parameter is the maximum TCP congestion window. A solution may consist exclusively of window-limited flows, which do not cause losses yet the aggregate throughput does reach the capacity of a target lossy link. To ensure that the link is in fact lossy, we impose additional constraints on the maximum congestion window configuration such that  $W > \sqrt{\frac{8}{3bp}}$  (flow is congestion-limited) [15] is satisfied for flows on lossy paths.

## 4 Evaluation

In this section, we will evaluate the proposed traffic generation algorithm. We refer to the proximity between the conditions that result from the generated traffic and the target link conditions as the *accuracy* of the traffic generation algorithm for a particular set of inputs. Of primary concern is to evaluate the accuracy and to examine the factors that cause significant inaccuracies.

For each set of user inputs, we instantiate and solve a linear program for the traffic matrix. We then configure and run *ns2* [3] simulations based on the user inputs and the traffic matrix solution. For each target link, we compare the observed utilization and/or loss rate against the target values. Suppose  $X_o$  and  $X_t$  are, respectively, the observed and target values of a link condition, we derive the *relative error*  $E$  for the link condition as follows:

$$E = \frac{X_o - X_t}{X_t}$$

In addition, we use the *relative error due to rounding* to assess the error incurred when rounding the solution to integers. Suppose that the load on a target link calculated from the original solution is  $L_t$ , and that the load calculated after rounding is  $L_i$ , then the relative error  $E_r$  due to rounding is:  $E_r = \frac{L_i - L_t}{L_t}$ .

### 4.1 Experimental Setup

Unless otherwise noted, the test inputs for our evaluation are described as follows:

- (a) We present results here based on two topologies. The first one is a GT-ITM transit-stub [5] *backbone* network with 200 nodes and 698 links, and the link capacities are uniformly distributed between 10Mbps and 100Mbps. The other topology has 55 nodes and 120 links, and it represents a network with both *backbone* and *access* links. The capacities of the backbone links are uniformly distributed between 10Mbps and 80Mbps. The access link capacity follows a discrete distribution, and takes on one of the following values: 1.5Mbps (30%), 10Mbps (50%), 100Mbps (20%).
- (b) We select target links randomly with a certain probability, which varies across experiments. The target utilization

for a non-lossy link follows a uniform distribution ranging from 0.2 to 0.8. The target loss rates on lossy links are uniformly distributed between 0.001 and 0.03.

(c) The maximum TCP congestion window size follows a uniform distribution between 15 and 45 packets. The TCP data segment size  $S$  is set to 1448 bytes.

(d) We use  $u = 0.1$ ,  $\alpha = 1.7$  and  $\beta = 1$  throughout the experiments, where  $\alpha$  and  $\beta$  define the Pareto distribution of the UDP ON/OFF intervals. We use  $H_t = 2.5$  unless otherwise noted.

### 4.2 Utilization Approximation

In this subsection, we demonstrate some of the factors that affect utilization approximation.

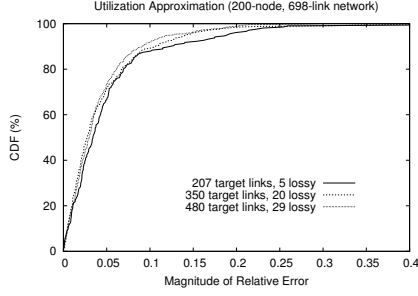
**Impact of Network Topology:** We did not observe a noticeable relationship between the accuracy of our algorithm and the scale and type of the input network topology. Due to space limitations, we do not show results for multiple network topologies. For Figure 2, we use the *backbone* topology as described in the experimental setup. Figure 2 shows the CDFs of the magnitude of  $E$  for three inputs with different numbers of target links and lossy links. For about 90% of the target links, the relative error is smaller than 0.1 in these experiments.

In the rest of this subsection, we use the *backbone* and *access* network topology described above to explore the impact of other factors.

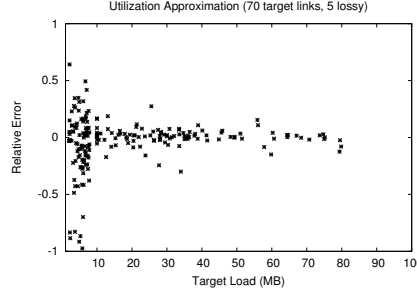
**Impact of Integer Rounding:** We first examine how much of the error in a typical setting is due to rounding.

Figure 3 shows the relative error for utilization constraints in experiments involving 70 target links, among which 5 are lossy. It reveals that the largest errors mostly occur on links with a small target load ( $Cp < 8Mbps$  in this case). The tendency of decreasing relative error with larger target load is consistent with the impact of rounding. In fact, we found that the distributions of  $E$  and of  $E_r$  are almost the same, indicating that rounding is almost solely responsible for the errors in this configuration.

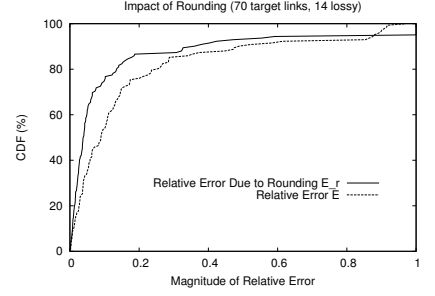
Figure 4 compares the magnitude of  $E$  and  $E_r$  for a configuration with 14 lossy links and 70 target links. On average, the error in this case is larger than what can be explained by rounding, especially for errors up to 0.3. This indicates that there are other factors contributing to the errors. In terms of the overall distribution, about 80% of the target links incur an error below 0.2. The slightly worse approximation, as compared to Figure 2, is explained by the fact there are access links in this topology with very low target loads (and hence very few flows traversing them), for which the impact of rounding could be significant.



**Figure 2. CDF of the Magnitude of Relative Error (backbone topology)**



**Figure 3. Relative Error vs. Target Load**



**Figure 4.  $E$  vs.  $E_r$  (14 lossy links, 70 target links)**

#### Impact of the Number of Target Links and Lossy Links:

We found that the accuracy of our algorithm is not noticeably affected by the number of target links, provided that the number of lossy links does not vary significantly. This is demonstrated by Figure 5, which shows the CDFs of  $E$  for four configurations that have different numbers of target links but a similar number of lossy links. Clearly, the CDFs are very close among the four configurations. Note that in Figures 5 and 6, we report results only for links with a relatively high target load ( $> 8$  Mbps), on which the errors are not dominated by  $E_r$ .

On the other hand, increasing the number of lossy links noticeably affects the overall accuracy. Figure 6 shows the CDFs of errors for four configurations with different numbers of lossy links and the same number of target links. Clearly, the average error increases with the number of lossy links. The impact of lossy links on other links is discussed at the end of the next subsection. We also found that the Average Flow Hop Count  $H_t$  bears on the accuracy, but *only when* there is a relatively large fraction of lossy target links.

#### 4.3 Loss Rate Approximation

We did not find the average loss rate approximation to be obviously input-dependent. Also, errors for the loss rate approximation span a larger range than utilization approximation. Figure 7 is a scatter-plot of the target loss rate vs. observed loss rate on different links, in the set of experiments presented in Section 4.2. For most of the samples, the ratio between the observed and the target loss rates is within the range  $[0.5, 3]$ .

One may think that the loss rate approximation error, at this level, would significantly affect the TCP throughput estimation, thus the utilization approximation of a lossy link. Fortunately, we find that the *average loss rate of TCP flows* is much closer to the target value than is the *average loss rate*. UDP flows, which are not adaptive to congestion, are

responsible for the larger-than-target average loss rate.

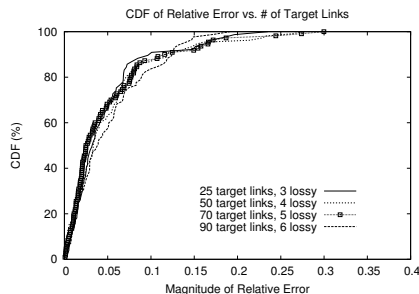
#### 4.4 Explaining the Impact of Lossy Links

Section 4.2 identifies lossy target links as one major factor that affects the accuracy of the utilization optimization. Lossy links affect the algorithm accuracy primarily because they might cause losses that are not presumed by the algorithm, as explained below.

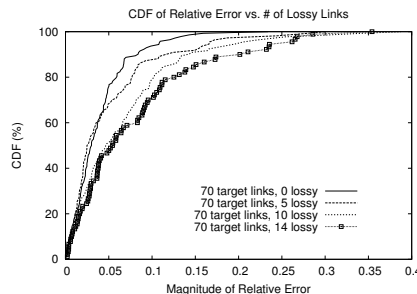
Our analysis of packet-level traces reveals that the largest errors take place exclusively on links of which the *reverse* direction is congested. In addition, the congestion in the reverse direction causes data packet losses in the forward direction of each of these links, which is not congested by its own TCP data load. This causes a significant error in the throughput estimate of a TCP data flow traversing the forward direction of the link. The cause of packet losses on such an uncongested link is the queuing of ACKs in the reverse direction, which leads to a phenomenon known as *ACK compression* [11]. It is, however, not easy to model the loss probability that will result from this phenomenon [11].

#### 4.5 Summary of the Evaluation Results

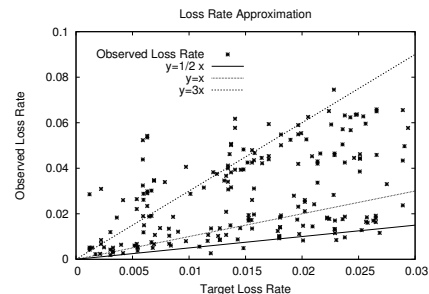
In this section, we have evaluated the accuracy of the proposed traffic generation algorithm in attaining the target link conditions. There are two major factors that affect the utilization approximation. First, rounding to integer numbers of flows can cause errors. Second, lossy target links could cause ACK compression, which eventually affects the estimate of loss rate for flows in the reverse direction; the impact of lossy links is more pronounced when there is more dependence, in terms of traffic, among links. In most of our experiments, the utilization error is smaller than 20% for as many as 80% of the target links, and rounding only affects links with a very small number (between 1 and 3) of flows. Loss rate accuracy is in general not as good, and the



**Figure 5. Impact of the Number of Target Links**



**Figure 6. Impact of the Number of Lossy Links**



**Figure 7. Loss Rate Approximation (scatter plot)**

ratio between the observed and the target values is mostly within the range  $[0.5, 3]$ . However, we believe that loss rate approximations in this range are acceptable to most target applications of the proposed traffic generation method.

## 5 Concluding Remarks

In this paper, we developed and evaluated a traffic generation algorithm for TCP simulations that matches the generated traffic to the network configuration such that the resulting link conditions are as specified by the user. The algorithm enables users to obtain realistic and explicitly controlled link conditions through background traffic simulation. This was achieved in a trial-and-error manner in previous work and it was possible only for very simple networks. While attaining target link conditions, our algorithm also minimizes the simulation memory requirement and conforms to certain realism considerations, such as the traffic mix and the average flow hop count. Our evaluations show that link conditions realized by the generated traffic are very close to the target conditions, and that the algorithm scales well with the size of the network topology and the number of target links. For future work, we intend to use the traffic generation algorithm to study in depth how sensitive different simulation studies are to certain characteristics of the background traffic. In addition, we will explore the application of this methodology with alternative traffic models or optimization objectives.

## References

- [1] <http://icir.org/models/trafficgenerators.html>.
- [2] D. Anderson, A. Snoeren, and H. Balakrishnan. Best-Path vs. Multi-Path Overlay Routing. In *IMC*, 2003.
- [3] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, and et al. Advances in Network Simulation. *IEEE Computer*, 2000.
- [4] P. Barford and M. Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *SIGMETRICS*, 1998.
- [5] K. Calvert, M. Doar, and E. Zegura. Modeling Internet Topology. *IEEE Communications Magazine*, 1997.
- [6] J. Cao, W. Cleveland, Y. Gao, K. Jeffay, F. D. Smith, and M. Weigle. Stochastic Models for Generating Synthetic HTTP Source Traffic. In *Infocom*, 2004.
- [7] A. Dhamdhere, H. Jiang, and C. Dovrolis. Buffer Sizing for Congested Internet Links (submitted for publication). In *Infocom*, 2005 (to appear).
- [8] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. *IEEE/ACM Transactions on Networking*, Dec 1997.
- [9] M. Grossglauser and J. Bolot. On the Relevance of Long-Range Dependence in Network Traffic. *ACM/IEEE Transactions on Networking*, 1999.
- [10] Y. Joo, V. Ribeiro, A. Feldmann, A. Gilbert, and W. Willinger. TCP/IP Traffic Dynamics and Network Performance: A Lesson in Workload Modeling, Flow Control, and Trace-driven Simulations. *Computer Communication Review*, 2001.
- [11] T. Lakshman, U. Madhow, and B. Suter. TCP/IP Performance with Random Loss and Bidirectional Congestion. *ACM/IEEE Transactions on Networking*, 2000.
- [12] K. Lan and J. Heidemann. Rapid Model Parameterization from Traffic Measurements. *ACM Transactions on Modeling and Computer Simulations (TOMACS)*, 2002.
- [13] L. Le, J. Aikat, K. Jeffay, and F. D. Smith. The Effects of Active Queue Management on Web Performance. In *SIGCOMM*, 2003.
- [14] M. Lucas, B. Dempsey, D. Wrege, and A. Weaver.  $(M, P, S)$ -An Efficient Background Traffic Model for Wide-Area Network Simulation. In *GlobeCom*, 1997.
- [15] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *SIGCOMM*, 1998.
- [16] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The End-to-End Effects of Internet Path Selection. In *SIGCOMM*, 1999.
- [17] M. Yuksel, B. Sikdar, K. Vastola, and B. Szymanski. Workload Generation for *ns* Simulations of Wide Area Networks and the Internet. In *CNDC*, 2000.
- [18] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An Information-Theoretic Approach to Traffic Matrix Estimation. In *SIGCOMM*, 2003.