TCP/IP traffic dynamics and network performance: A lesson in workload modeling, flow control, and trace-driven simulations

Youngmi Joo*, Vinay Ribeiro†,
Anja Feldmann‡, Anna C. Gilbert, and Walter Willinger§

This paper is dedicated to the memory of our friend and colleague, Youngmi Joo, who died on April 18th, 2001 at Stanford University.

ABSTRACT

The main objective of this paper is to demonstrate in the context of a simple TCP/IP-based network that depending on the underlying assumptions about the inherent nature of the dynamics of network traffic, very different conclusions can be derived for a number of well-studied and apparently well-understood problems in the area of performance evaluation. For example, a traffic workload model can either completely ignore the empirically observed high variability at the TCP connection level (i.e., assume "infinite sources") or explicitly account for it with the help of heavy-tailed distributions for TCP connection sizes or durations. Based on detailed *ns-2* simulation results, we illustrate that these two commonly-used traffic workload scenarios can give rise to fundamentally different buffer dynamics in IP routers. Using a second set of *ns-2* simulation experiments, we also illustrate a qualitatively very different queueing behavior within IP routers depending on whether the traffic arriving at the router is assumed to be endogenous in nature (i.e., a result of the "closed loop" nature of the feedback-based congestion control algorithm of TCP) or exogenously determined (i.e., given by some conventional traffic model – a fixed "open loop" description of the traffic as seen by the router).

*Y. Joo was with the Department of Electrical Engineering, Stanford University, CA 94305-9030.

†V. Ribeiro is with the ECE Department, Rice University, Houston, TX 77005, email: vinay@rice.edu.

‡A. Feldmann is with the Universität des Saarlandes, Saarbrücken, Germany, email: anja@cs.uni-sb.de.

§A. Gilbert and W. Willinger are with AT&T Labs-Research, Florham Park, NJ 07932-0971, email: {agilbert,walter}@research.att.com. W. Willinger's research was partly supported by the NSF Grants NCR-9628067 at the University of California at Santa Cruz and ANI-9805623 at Boston University.

1. INTRODUCTION

Traffic characterization and modeling are generally viewed as important first steps toward understanding and solving network performance-related problems. At the same time, there is little disagreement that the resulting understanding of and solutions to network performance problems are only as good and complete as the underlying assumptions on the usage of the network and the nature of the traffic that it carries.

The main goal of this paper is to highlight the extent to which assumptions underlying the nature of network traffic can influence practical engineering decisions. More specifically, using a toy example of a TCP/IP network and relying on the *ns-2* network simulator [2], we illustrate how by either implicitly accounting for or explicitly ignoring some aspects of the empirically observed variability of network traffic, a range of different and at times opposing conclusions can be drawn about the inferred buffer dynamics for IP routers. While there are many known causes for the observed variability in measured TCP/IP traffic, in this paper we focus on just two of them (for more details, see [5]). On the one hand, we explore contributions due to TCP connection-level variability – heavy-tailed TCP connections are a known cause for the empirically observed self-similar scaling behavior of aggregate packet traffic over large time scales (i.e., larger than RTT time scales). On the other hand, we consider an aspect of traffic variability that concerns the flow of packets within individual TCP connections – TCP's feedback-based congestion control algorithm is a suspected contributing factor to the observed variability of measured TCP/IP traffic over small time scales (i.e., on the order of RTT).

As far as large time scale or TCP connection-level variability is concerned, the extreme case of no variability can be achieved by requiring at the application or connection layers that each active source has to transfer an essentially infinite file for the duration of the entire simulation. This *infinite source model* lacks the natural variability that has been observed in measured data at the application/connection level and, in turn, lacks any non-trivial large time scale variability in the corresponding rate process [17, 4, 7]. In contrast, large time scale variability that is consistent with self-similar scaling over those time scales and has become a trademark of measured traffic rate processes can be achieved in a parsimonious manner by explicitly accounting for an adequate level of variability at the application/connection layer [21, 20]. In fact, by replacing the infinite source model by a SURGE-like workload generator [3], that is, by changing an infinite file transfer into an application that imitates a typical Web session (with appropriately chosen heavy-tailed distributions to match the observed variability of various Web-related items such as session length and size, size of requested Web pages), we obtain a *Web-user source model* that automatically generates the desired large time scale variability in the aggregate rate process. These workload models are described in more details in Section 2.

In the first set of *ns-2* simulation experiments, we demonstrate in Section 3 how the two source models lead to qualitatively very different queueing dynamics in the router. While the queueing behavior with the infinite source model shows pronounced periodic fluctuations (commonly referred to as "synchronization effects"), these effects essentially disappear when the sources are Web users. In the context of single bottleneck link networks and infinite source models, TCP/IP traffic "synchronization effects" have been well studied in the past and have led to many influential papers on understanding the dynamics of TCP-type congestion control algorithms, [14, 18, 22, 23]. We contribute in this paper to the existing body of knowledge by demonstrating how accounting for realistic workload heterogeneity (via appropriate TCP connection level variability) can yield new insights into actual TCP behavior. In fact, using the terminology originally due to V. Jacobson, we show how the presence of many short-lived TCP connections or "mice" – a salient feature of our Web workload model – completely changes the buffer dynamics that has been noted in the past when the network load is entirely due to the long-lived TCP connections or "elephants" of the infinite source model.

In a second set of *ns-2* simulation experiments, we focus in Section 4 on the role of traffic variability that is due to TCP's feedback flow control mechanism and that partly determines the flow of packets emitted from the different sources during each TCP connection. To this end, we perform a number of related *closed loop* and *open loop* simulations and compare them on the basis of some commonly-used performance criteria. Here, by "closed loop" we mean a *ns-2* simulation with a fixed networking configuration, including buffer size in the router(s), link bandwidths, delays, etc., and where all hosts use TCP. Note that because of its closed loop nature, TCP constantly shapes (with some delay) the packet flow emitted from the different sources, which in turn alters the rate process that arrives at the IP router for buffering (which in turn impacts the level of congestion, etc.). In contrast, "open loop" means we collect a packet trace from a particular *ns-2* simulation run (or alternatively, from a link within the Internet) and use it to perform a set of *trace-driven* simulations of a queueing system that represents our IP router. Clearly, trace-driven simulations cannot account for the capabilities of the network to constantly shape and thus alter the offered traffic to the queue. Although the "open loop" vs. "closed loop" effects on buffer dynamics seem to be part of the networking "folklore," they are not well documented in the literature (for a noticeable exception, see the recent paper [1] by Arvidsson and Karlsson, who provide a critical assessment of traffic models for TCP/IP by comparing "live" or closed loop scenarios with the corresponding "trace" or open loop counterparts). In this sense, our contribution in this paper consists of contrasting the "open loop" vs. "closed loop" approach in a simple context and demonstrating that inferring performance of a closed-loop system based on an analysis of its open loop counterpart can give rise to irrelevant results and lead to misleading conclusions.

We conclude in Section 5 by commenting on a number of admittedly unrealistic assumptions regarding our toy network simulation models. Clearly, the special appeal of these toy models is that they can serve as reasonable starting points for understanding TCP dynamics in large-scale complex topologies under realistic traffic scenarios, but they are certainly not meant to faithfully model real-world environments. Therefore, one of their primary purposes should be their ability to detect, identify, or point out those aspects of a real-world environment (e.g., the presence of many short-lived TCP connections, TCP feedback control, network topology) that are likely to play a crucial role in gaining a solid understanding of how modern TCP/IP-based networks such as the Internet work. We believe that the findings reported in this paper will increase the overall awareness that it is in general easy to draw conclusions based on infinite source models and/or open loop systems and their performance in a simple setting, but that the real challenges lie – as succinctly discussed in [11] – in convincingly demonstrating that these conclusion either still hold or become invalid for realistically loaded real-world networks and/or the corresponding closed loop systems and their performance.

2. THE SIMULATION SETUP

In this section, we give a description of the networking configuration used throughout this paper and spell out the details of our workload models. In addition, we discuss the simulation engine used for our studies and review some TCP-specific features that are necessary to present and discuss the findings of our simulation experiments.

2.1 A simple networking configuration

All of the simulation experiments reported in this paper involve the simple network topology depicted in Figure 1. Our toy network consists of a single server (node 1), a set of low-speed clients (nodes $5-405$), and a number of links. The clients are connected to the network via $40-100$ Kbps links, the server has a 100 Mbps connection to the network, and two additional links (A, B) comprise the rest of the network. Link A is used to limit the capacity of the network to a value between 1.5 Mbps and 3 Mbps and represents, in fact, the bottleneck link in our simulation setup. The link B bandwidth is set at 100 Mbps, and we use this link to observe the dynamics of the traffic before it traverses the bottleneck link A. To gauge the impact of the buffer size in the router, we vary the number of buffer spaces available for buffering packets[1] in node 3 for link A; depending on the particular experiment, this number can vary anywhere from 10 to 1000. If the router's buffer is full, it deterministically drops a newly arriving packet ("drop tail"). All simulations are based on "drop tail," with the exception of two experiments (see Section 3), where we will consider the "random

[1] *Ns-2* allocates buffer space in terms of number of packets and not number of bytes.

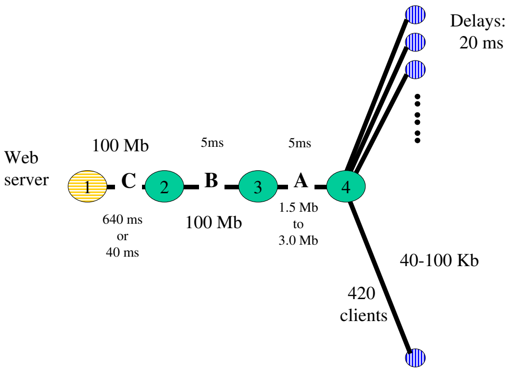


Figure 1: Network configuration

early detection" or RED algorithm for dropping packets [8, 9].

Note that this network configuration allows for no variability as far as delays, round-trip times and cross traffic are concerned. While all of these three aspects of realistic network traffic are known to be crucial for matching the variability inherent in measured Internet traffic [5], our focus in this paper is on an extreme case of networking homogeneity, where there is a single bottleneck through which all the traffic has to go and where all packets experience one and the same delay in the network, namely 1.4 seconds (to compare, we also experiment with a link delay of 0.14 seconds). We chose this admittedly unrealistic network configuration because we wanted to illustrate our findings in a setting that has been used (with different parameter values, though) in a number of previous studies of the dynamics of TCP (e.g., [18, 22, 23]).

2.2 Two different workload models

In contrast to our homogeneous network configuration, our workload models for the clients are heterogeneous in nature and allow for a spectrum of variability at the user level. On the one side of the spectrum, we deal with the case of *no* user level variability by considering 50 *infinite sources* that always have data to transfer for the duration of the entire simulation (that is, 4200 seconds). To eliminate any transient effects, the sources start their transfers at random times – picked according to a uniform distribution – during the first 600 seconds of the simulation, and when analyzing the output of our simulations, we focus on the remaining 3600 seconds of each simulation run when all 50 source are known to be active.

To cover the other side of the spectrum, we rely on a *Web workload* model considered in [5] that is very similar to SURGE developed at Boston University [3]. The main idea behind these Web workload models is that during a Web session, a user typically requests several Web pages, where each Web page may contain several Web objects (e.g. `jpg` images or `au` files). To parsimoniously capture the observed variability within the different layer of this natural hierarchical structure of a typical Web session (e.g., see [4, 7, 3]), we consider certain types of probability distribution functions for the following session attributes: number of pages per session, inter-page time, number of objects per page, inter-object time, and object size (in KB). The specifics of these distributions, including the corresponding parameter values, are given in Table 1. Note that the empirically observed high variability associated with these Web session attributes is naturally captured via Pareto-type distributions with appropriately chosen parameter values.

In practice, a Web session will request a set of pages (usually 300) in the following manner.[2] After the completion of the download of all objects in the last page, it will wait for a random amount of time (sampled from the inter-page time distribution) before requesting the download of the next Web page. At the start of the next Web page download, the client determines the server, which in our setup is always node 1. In addition, the client chooses the number of objects in the next Web page by picking a random number according to the objects-per-page distribution. For each object, the client determines the size of the object (by sampling from the object size distribution) and then sends a request to the server to download the object. The time between the request for two different objects within a Web page is chosen according to the inter-object-time distribution. Once all objects of a Web page have been downloaded, the process repeats itself, i.e., after a waiting time (sampled from the the inter-page-time distribution), the next Web page is downloaded, etc.

In our simulations, we use 350 Web sessions, all of which start at a random point in time within the first 600 seconds of each simulation run. Also note that the infinite source model can be viewed as a special case of our Web workload model, where the number of pages per client is 1, the number of objects per page is 1, and the object size is set to a very large value (e.g., 10000000) to ensure that the client does not run out of data to send for the duration of the simulation. In this sense, Table 1 provides a complete specification of the two workload models used in our experiments below. It is also worth pointing out that our two workload models are comparable in the sense that, for example, for a typical RTT on the order of 1 second, the offered load (i.e., the load measured on link A for the same configuration as shown in Figure 1, but where A is no longer the bottleneck) generated by the 50 infinite sources is about 66% of that generated by the 350 Web sources; even though it needs about 75 infinite sources to generate approximately the same offered load, the results discussed in this paper are largely insensitive to such differences in the offered load.

2.3 Some TCP details – for later reference

The simulation engine used throughout this study is *ns-2* (Network Simulator version 2) [2]. This discrete event simulator provides a rich library of modules such as UDP, different flavors of TCP, buffer scheduling algorithms, routing mechanism, and trace collection support. For the purposes of this paper, we rely on the fact that *ns-2* comes

[2]Note that in a typical HTTP 1.0 transaction, a Web client sends a request to the Web server for a Web object after establishing a TCP connection. The Web server responds with a reply header and then continues to send the data. To circumvent some limitations in the original *ns-2* TCP connection module we emulated the exchange of the HTTP header information with two TCP connections.

Name	number	inter–page	objects/page	inter–object	object size
INFINITE SOURCE	Constant 50	Constant 1	Constant 1	—	Constant 10000000
WEB SOURCE	Constant 350	Pareto mean 50 shape 2	Pareto mean 4 shape 1.2	Pareto mean 0.5 shape 1.5	Pareto mean 12 shape 1.2

Table 1: Summary of the relevant distributions (with parameter values) for the two workload used in our simulation experiments.

with a thoroughly-checked and well-studied implementation of TCP Reno, the most widely used version of TCP today, and in the following, we list and discuss some of the protocol-specific details that relate directly to what is presented and explained in the subsequent sections. For a comprehensive treatment of the different versions of TCP, see for example [19].

One of the main themes in Sections 3 and 4 below will be how TCP exploits available network resources (e.g., limited buffer space in end-systems, or limited bandwidth along end-to-end paths). To this end, recall that TCP provides end-to-end flow control using a sliding window protocol. As far as a buffer-limited receiver is concerned, it sends an *advertised window* informing the sender how many segments it can have in flight beyond the latest one acknowledged by the receiver. An important property of a sliding window protocol is that it leads to *self-clocking*; that is, no matter how fast the sender transmits, its data packets will arrive at the receiver spaced out in a way that reflects the network's current carrying capacity. In addition, to control how the sender attempts to consume the available network path capacity, TCP maintains a congestion window, or CWND. At any time, the sender can transmit up to the minimum of the advertised window and CWND. At the start of a TCP connection, the advertised window is initialized to the buffer size of the receiver, and the congestion window is limited to one or two segments. Each received acknowledgment, unless it is a duplicate acknowledgment, is used as an indication that data has been transmitted successfully and allows TCP to move the advertised window and to increase CWND. However, increasing the congestion window depends on the state of the TCP connection. If the connection is in *slow start*, it is increased exponentially per round-trip time (RTT), or more precisely, by one segment for every acknowledgment. If the connection is in *congestion avoidance*, it is increased linearly by one segment per RTT. TCP switches from slow start to congestion avoidance if the size of the congestion window is equal to the value of a variable called the *slow start threshold*, or SSTHRESH, for short.

Another important functionality provided by TCP is reliability. To insure that each segment is transmitted reliably, TCP – among other precautions – maintains a *retransmission timer*, or RTO, for the different segments. If no acknowledgment for the segment is received by the sender from the receiver within the timer period RTO, TCP assumes that the segment has been lost and retransmits it. Another way for TCP to detect losses is based upon duplicated acknowledgments. Since acknowledgments are cumulative, and since every segment that is received out of order (i.e., non-consecutive segment numbers) triggers an acknowledgment, TCP assumes that four duplicated acknowledgments indicate that a segment was lost. As before, TCP in this case retransmits a segment if it detects a lost segment. If TCP detects a packet loss (which in today's Internet is interpreted as an indication of congestion), either via timeout or via duplicated acknowledgments, SSTHRESH is set to half of the minimum of CWND and the advertised window size. If the loss was detected via timeout, the congestion window is set to one segment, and TCP returns to slow start; note that in this case, the self-clocking pattern has been destroyed and entering slow start offers the potential for rapidly re-discovering the network's current carrying capacity. In contrast, if the loss was detected via duplicate ACKs, then the self-clocking pattern is still present, and TCP does not cut back the rate as drastically as before, though it immediately enters congestion avoidance.

Finally, we will also consider different strategies for managing the router's buffer at node 3 for link A. For the majority of simulation experiments conducted in this paper, we employed the "drop tail" strategy which deterministically drops a newly arriving packet whenever the router's buffer is full. To contrast, for two simulation experiments discussed in Section 3, we considered the "random early detection", or RED strategy [8, 9]. Note that in the case of RED, whenever an arriving packet sees that the buffer is filled to more than a certain fraction of its total size, the packet is dropped by the router with a certain probability.[3]

3. IMPACT OF VARIABILITY AT THE APPLICATION LAYER

In this section we demonstrate how the TCP flow control algorithm can lead to artifacts in the router buffer dynamics at node 3, depending on which of the two workload models are used to generate the network traffic – only long-lived TCP connections or "elephants" or a mixture of "elephants" and short-lived "mice."

3.1 No variability and the occurrence of periodic fluctuations

We first consider the case where 50 clients generate traffic according to the infinite source workload model. For a randomly selected 40 second long portion of our simulation, the top plot in Figure 2 shows the traffic rate process (i.e., number of packets per second) as it arrives at the queue at node

[3]While the RED simulations in this paper could benefit from a more careful choice of the underlying parameter settings [12], we consider here only RED simulations, where the parameter settings are the *ns-2* default values; namely THRESH=5 and MAXTHRESH=15.

3, to be forwarded through the bottleneck link A to the different destinations, and the instantaneous buffer occupancy of the queue at node 3. Here, the link C delay is 640 milliseconds, the maximum buffer occupancy of the queue at node 3 is assumed to be 50 packets, the bottleneck link A has a bandwidth of 1.5 Mbps, and the queueing discipline is "drop-tail." Each packet drop is viewed by TCP as an indication of network congestion and results in a signal back to the sender of the dropped packet to reduce its sending rate, which in turn reduces the overall packet rate process arriving at the node 3 queue. This feedback dynamic is an inherent feature of TCP's end-to-end flow control mechanism, and its effect, in the presence of infinite sources, shows up as pronounced periodic fluctuations in the aggregate packet rate process and the queue occupancy process. To explain, shortly after the queue is filled and packets are dropped, the rate process stops increasing and drops after some time from a maximum of around 215 packets per second to about 150 packets per second. Such a drastic reduction in the rate process arriving at the queue allows the queue to completely drain its buffer. Upon experiencing no dropped packets, the individual TCP connections start to increase their sending rates again, which in turn results in an increase of the overall packet rate process (from about 150 to 215 or so packets per second) as seen by the queue. As a direct result of this higher arrival rate, the buffer at the node 3 queue starts to fill up again. Once the buffer is full, another round of packet drops will cause the affected connections to again reduce their sending rates, and the same process as before repeats itself. These predominant periodic fluctuations of the packet rate process and the buffer occupancy process are fully consistent with some of the original studies of the dynamics of TCP as described, for example, in [18] [23], where the term "synchronization" was used to describe these periodic fluctuations.

To demonstrate that these pronounced periodic fluctuation for the packet rate and buffer occupancy processes are not an artifact of the extremely large delay value of 640 milliseconds for link C, we show in the bottom plot of Figure 2 the results for of the same simulation experiment, except that the link C delay is now reduced to 40 milliseconds. Note that we still observe pronounced periodic behavior, but since the feedback to the clients is now much more immediate, the queue does not have time to drain completely, the cycle length is significantly shorter, the overall link utilization is larger, and the rate process is somewhat less variable. Nevertheless, the arguments for the observed periodic fluctuations remain the same as before.

Figure 3 shows the results of the same simulation experiments as in Figure 2, but with the "drop tail" buffering strategy replaced by "random early detection" or RED mechanism described in Section 2. Even though this change in how packets are dropped at the router's buffer introduces a certain amount of randomness, Figure 3 illustrates that this additional randomness is not strong enough to significantly alter the periodic fluctuations in Figure 2. However, RED clearly succeeds in maintaining smaller queues in the buffer as compared to using the "drop tail" buffer management strategy.

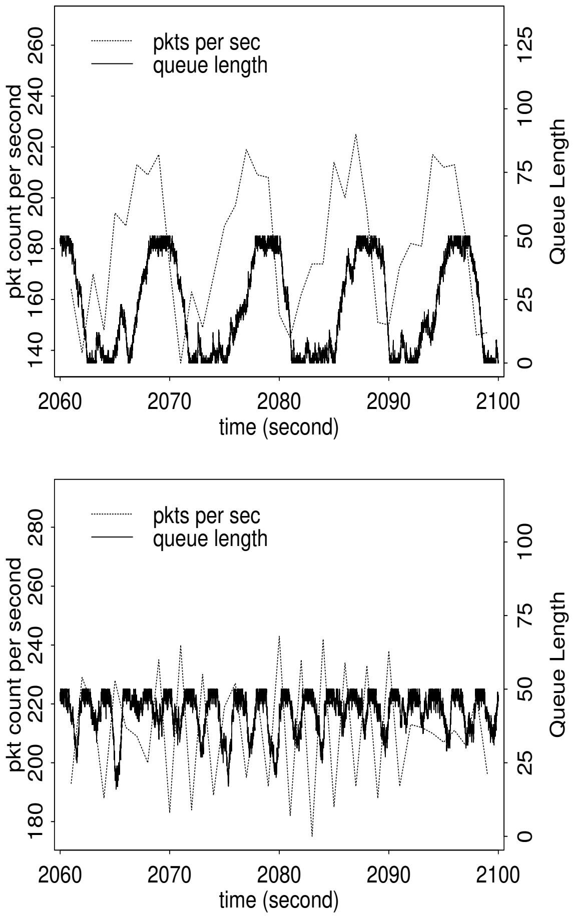


Figure 2: Packet rate process (dotted line) and buffer occupancy process (solid line) under "drop tail" (top: 50 infinite sources, link C delay 640 milliseconds; bottom: 50 infinite sources, link C delay 40 milliseconds).

3.2 High variability and the absence of periodic fluctuations

To demonstrate how allowing for realistic variability at the application level changes the qualitative features of Figure 2, we replace the infinite sources with clients that generate traffic according to our Web workload model. Intuitively, while retaining a few "elephants," this source model ensures that a significant portion of the TCP connections are short-lived (i.e., representing many "mice"). The results are shown in Figure 4, where we again depict the packet rate and buffer occupancy processes for a 40 second portion of our simulations for the case of a 640 milliseconds (top plot) and 40 milliseconds (bottom plot) link C delay, respectively. In stark contrast to the case of infinite sources and the resulting periodic fluctuations, the inherent variability of the TCP connections in the case of the Web sources gives rise to packet rate and buffer occupancy processes that completely lack any sort of periodic components.[4]

[4]In addition to relying on visual means for checking for the absence or presence of pronounced periodic components in

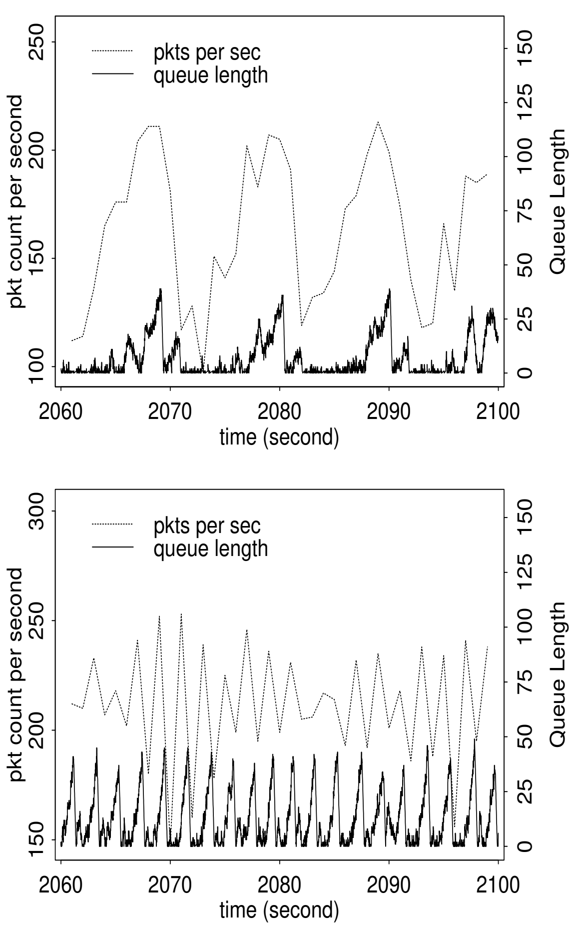


Figure 3: Packet rate process (dotted line) and buffer occupancy process (solid line) under RED (top: 50 infinite sources, link C delay 640 milliseconds; bottom: 50 infinite sources, link C delay 40 milliseconds).

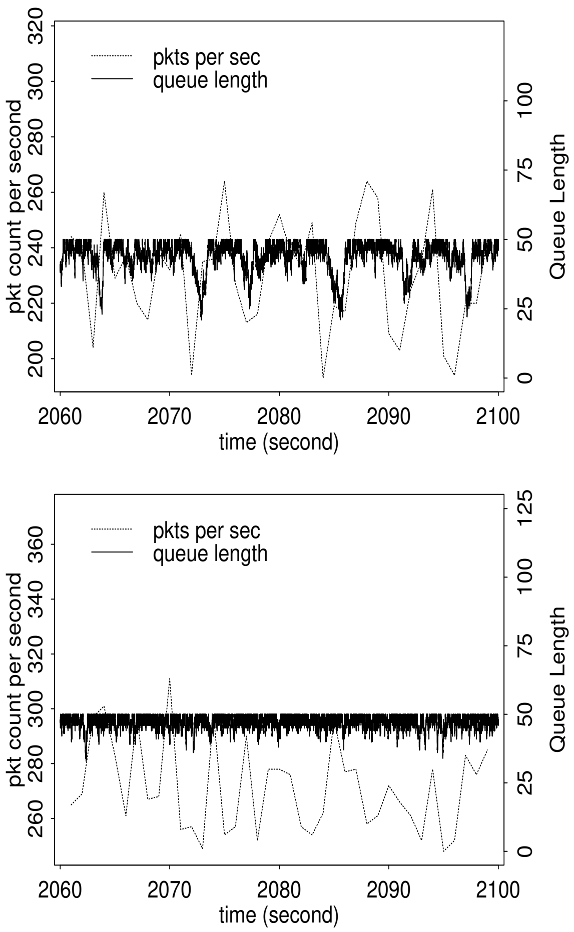


Figure 4: Packet rate process (dotted line) and buffer occupancy process (solid line); top: 350 PARETO 1-type Web clients, link C delay 640 milliseconds; bottom: 350 PARETO 1-type Web clients, link C delay 40 milliseconds.

The difference between the two workload scenarios is especially striking for the case of a link C delay of 640 milliseconds. While in the presence of many "mice", the buffer never has a chance to completely drain, it does so on a regular basis for the "elephants-only" case. Also note that while in the case of the Web sources, the packet rate process arriving at the node 3 queue never drops below 190 or so, it drops to about 140 in the case of infinite sources. Together, these effects result in a significantly higher overall utilization when there exists a proper mixture of "mice" and "elephants." In the "elephants-only" case, even if we were to increase the number of long-lived connections, because of the presence of periodic fluctuations caused by the interactions between workload and TCP feedback, the overall link utilization would not increase significantly.

the buffer occupancy process, we also used more quantitative methods (e.g., straight-forward frequency domain-based techniques) to confirm our qualitative findings.

3.3 On why "mice" can get rid of periodic fluctuation

In contrast to the infinite source model, our Web workload model – via its built-in Pareto-type distributions for the different Web session attributes – guarantees that a significant amount of TCP connections are very small and hence short-lived. Although in today's Internet, the "elephants" are responsible for a major portion of the overall workload (i.e., number of bytes), the total number of packets due to the "mice" generates sufficient traffic to create losses at random points in time. This feature and the fact that the arrival patterns of the "mice" tend to be highly bursty (e.g., see [6]) suggest that the presence of significantly many "mice" makes it nearly impossible for the traffic rate and buffer occupancy processes to exhibit pronounced periodic fluctuations.[5] In this subsection, we explain why the presence of

[5]Another known cause that works against the presence of periodic features in real network traffic is the observed variability in round-trip time [5], but this cause is not accounted for in our simulation setup.

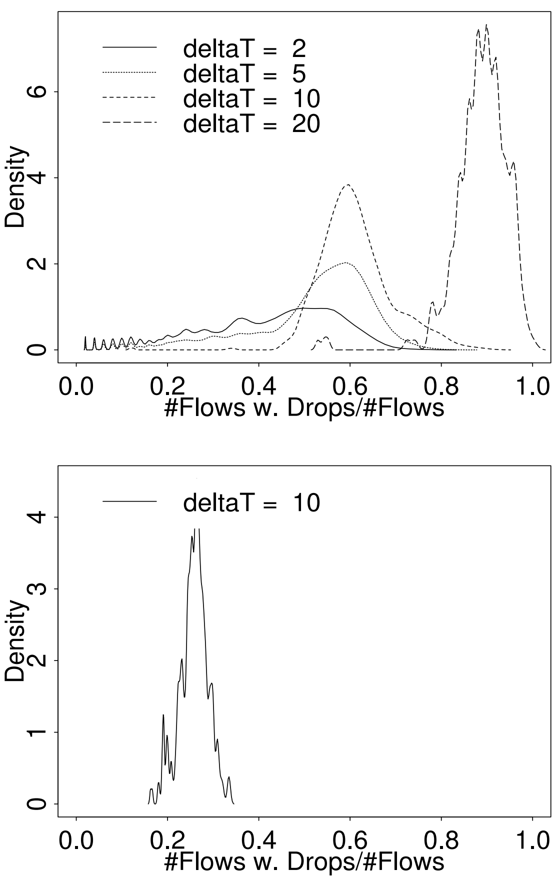


Figure 5: Effect of synchronization: Fraction of connections that experience packet losses (top: 50 infinite sources, link C delay 640 milliseconds, "drop tail"; bottom: 350 Pareto 1-type Web clients, link C delay 640 milliseconds, "drop tail").

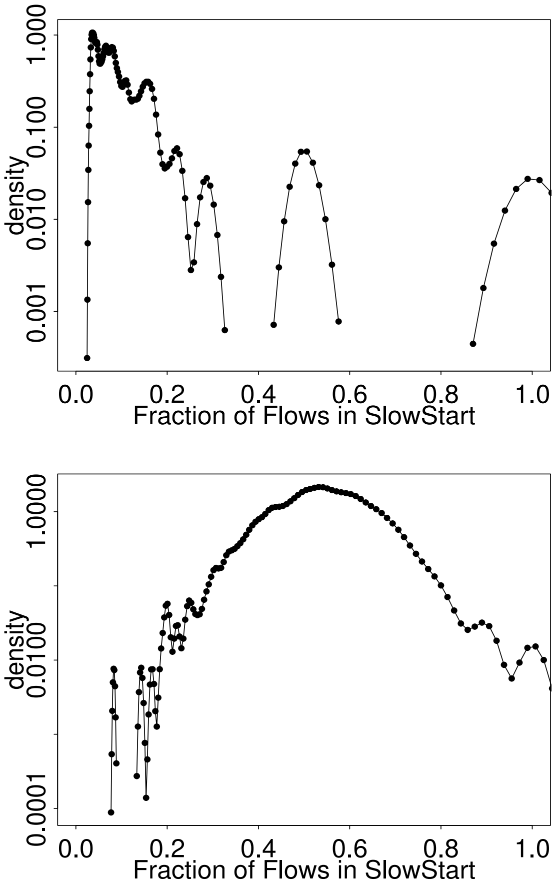


Figure 6: Effect of synchronization: Fraction of lossy connections that were in slow-start at the time of packet loss (top: 50 infinite sources, link C delay 640 milliseconds, "drop tail"; bottom: 350 Pareto 1-type Web clients, link C delay 640 milliseconds, "drop tail").

short-lived flows essentially gets rid of the periodic fluctuations that are encountered when assuming infinite sources. To this end, we consider our Web workload model and provide detailed information about how individual connections are affected by and/or react to packet drops. In terms of network configuration, we discuss in the following the case where the link C delay is 640 milliseconds.

In order to gauge the difference in the way congestion (i.e., packet drops) affect flows, we check what percentage of connections (out of all connections) experiences a loss of (at least) one packet during a time interval of size ΔT. Figure 5 shows the densities of these fractions for our two different workloads models and for different values of ΔT. Note that for the infinite source model (top plot), as ΔT increases from 1 second to 5 and 10 seconds, the densities become more centered around 60%, and when ΔT is increased even more to 20 seconds, the corresponding density function shifts toward 100%, with a mean larger than 90%. Given that the buffer occupancy process in Figure 2 (top plot) has a periodicity of about 10 seconds, we can conclude that about 60% of all connections lose at least one packet within this period and that almost every connection loses at least one of its packets within two such periods.[6] In contrast, for the Web workload model (bottom plot), the density functions corresponding to the different ΔT values turn out to be essentially identical to one another (the plot only shows the density corresponding to $\Delta T = 10$), and they are all sharply concentrated around 25%. Note that "mice" can start when other connections are reducing their sending rates, and they often finish before they experience any dropped packet. This also explains why the fraction of connections (out of all connections) experiencing packet drops is significantly smaller in the presence of "mice" than in their absence.

For another way to illustrate how the presence of many "mice" manifests itself in the observed TCP dynamics, we

[6] Thus, even if a connection managed to avoid dropping a packet during one congestion epoch, it is almost certain to experience a packet drop during the subsequent cycle. The fact that in the infinite source model, every TCP connection is almost certain to experience a packet drop suggests that each TCP connection receives a reasonable share of the bottleneck bandwidth; that is, the simulation setup is not likely to exhibit the traffic phase effects described in detail in [10].

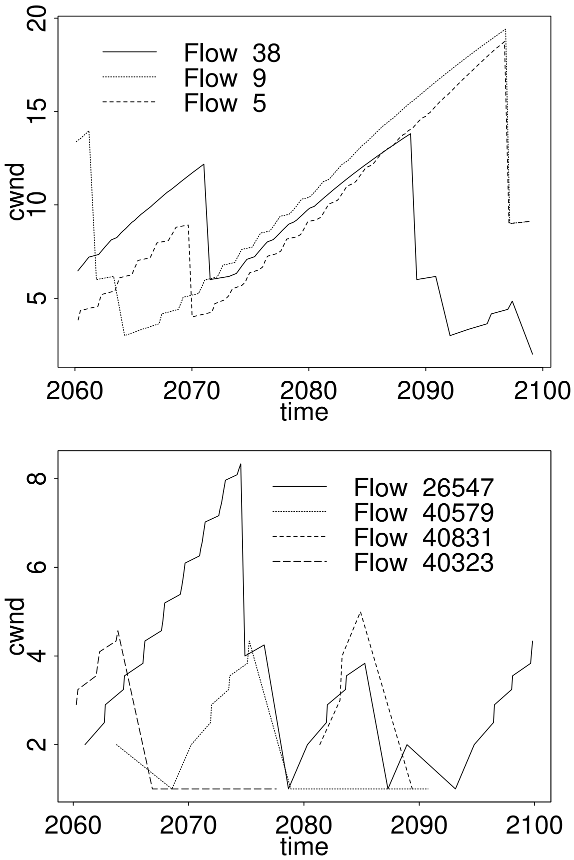


Figure 7: Dynamic of congestion window for a few selected flows (top: 50 infinite sources, link C delay 640 milliseconds, "drop tail"; bottom: 350 PARETO 1-type Web clients, link C delay 640 milliseconds, "drop tail").

consider all connections that had (at least) one of their packets dropped and depict in Figure 6 the fraction $0 \leq p \leq 1$ of those connections that were in slow-start at the time of packet drop (note that $1 - p$ gives the fraction of connections in congestion avoidance); the top plot is for the infinite source model, the bottom plot shows the same information for the Web source model.[7] The differences between the plots are again telling. In the presence of many "mice" (bottom plot), most of the connections were in slow-start at the time of packet drop, meaning that they were either very short or experienced multiple dropped packets; a closer look (see Figure 7, bottom plot) reveals that both cases usually contribute to this dominant slow-start effect. In general, these connections detect congestion via timeout (thereby losing the self-clocking pattern) which is less efficient than fast retransmit. Intuitively, with many of the connections going through slow-start and thereby increasing their bandwidth usage much more aggressively compared to those that are in congestion avoidance, the Web sources generally succeed via the presence of many "mice" to claim any unused bandwidth and utilize the network resources efficiently. In contrast, the top plot in Figure 6 shows that in the absence of any "mice," lots of the affected connections are in congestion avoidance and increase their bandwidth usage more gradually. Indeed, if we consider the size of the congestion window when losses occur for the infinite source case (see Figure 7, top plot), it tends to be significantly larger than when the sources generate traffic according to our Web workload model.

These observations also suggest that the dynamics of packet drops may be qualitatively different for the two workload scenarios. To confirm this conjecture, we consider in Figure 8 the distribution of the number of consecutively dropped packet (for the aggregate packet stream) for the infinite source and Web source models. As expected, the infinite source model results in a distribution that implies a less bursty drop process than for the Web source model. To explain, "elephants" are more likely to be in congestion avoidance than in slow-start, which means that they can only send one packet for every received acknowledgment. In contrast, we have seen that the presence of many "mice" results in many connections being in slow-start, which in turn increases the likelihood that more than one packet of a given flow can be dropped within a short period of time. In other words, while the loss dynamics induced by TCP when there is no variability at the application layer results in small bursts of consecutive packet drops, these bursts can be significantly larger when allowing for application-layer variability in accordance with our Web workload model.

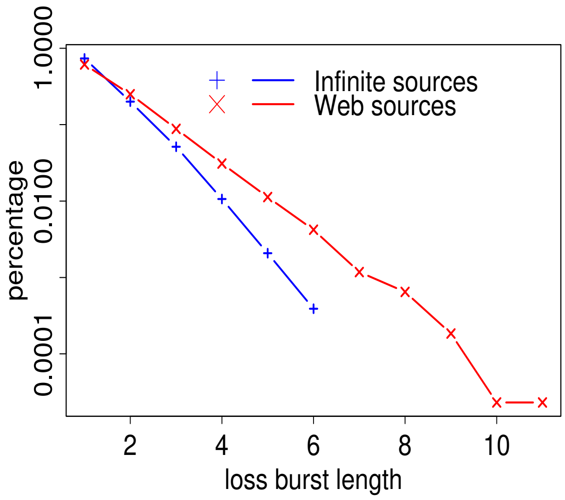


Figure 8: Dynamics of packet losses: distribution of number of consecutive packet losses for the aggregate packet stream.

To summarize, our simulation experiments have demonstrated that the dynamics of TCP can interact in intricate ways with the dynamics of the underlying workload model for the sources. Static workloads such as infinite sources allow for no variability at the source/connection level and are destined to produce pronounced periodic fluctuations in the packet rate and buffer occupancy processes. Even changing the queueing discipline at the router from "drop tail" to a simple version of RED does not appear to be sufficient to eliminate this phenomenon. However, as soon as

[7] These percentages are calculated for successive time intervals of length two seconds, ignoring intervals without any dropped packets. Different choices for the length of the time interval yield essentially identical plots.

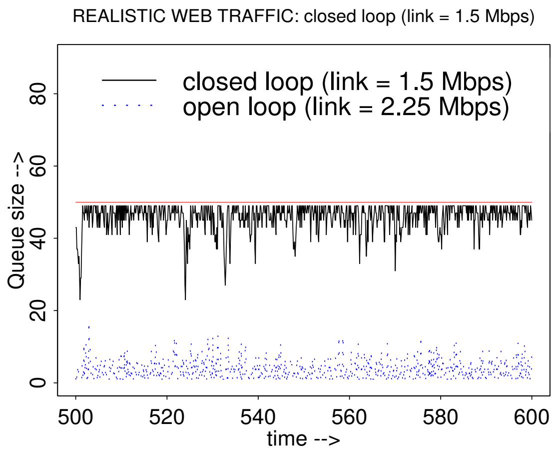
the workload model accounts for sufficient variability at the source/connection level, the "elephants" are forced to compete with many "mice" for the available network resources. The resulting heterogeneity in TCP states is sufficient to eliminate any potential for periodic fluctuation, and, as a result, the available resources are used more efficiently. In contrast, periodic fluctuations generally lead to lower link utilization, less bursty losses, and more homogeneity in terms of TCP states.

4. ON THE IMPACT OF FEEDBACK FLOW CONTROL

In the previous section, we used a set of *ns-2* simulation experiments to demonstrate that drastically different buffer dynamics in routers can arise, depending on the assumed nature of variability exhibited at the application/connection level. Our second set of *ns-2* simulation experiments is intended to increase the awareness within the performance modeling community that traditional performance evaluation (consisting of either assuming a given model for describing the traffic arriving at the queue or by performing trace-driven simulations) may have little or nothing to say about user-perceived end-to-end performance in the Internet, where a dominant fraction of traffic is governed by feedback control. Note that feedback constantly shapes and changes the packet flows emitted from the different sources, which in turn alters the rate processes that arrive at the IP routers for buffering, which in turn impacts the levels of congestion, etc.

To this end, we performed a number of related *closed loop* and *open loop* simulations and compared them on the basis of some commonly-used performance criteria. Here, by "closed loop" we mean a *ns-2* simulation with a fixed simple topology, including buffer size in the router(s), link bandwidths, delays, etc. and where all hosts use TCP; that is, for a given workload model at the source level, the TCP/IP protocol suite is used to exchange data between the different hosts, which in turn determines the flow of packets over the different links within the network. In contrast, "open loop" means that we collect a packet trace from a particular *ns-2* simulation run and use it to perform trace-driven simulations of a queueing system that represents our IP router. On the one hand, running trace-driven or open loop simulations is like assuming UDP-type sources in the sense that there is no feedback between the network and the sources generating the traffic; on the other hand, trace-driven or open loop differs from UDP because the timing of packet arrivals at the router is determined by the trace at hand and is generally not regular as is the case with UDP. Note that by their very nature, open loop or trace-driven simulations cannot account for the capabilities of the network to shape and thus alter the offered traffic to the queue (e.g., as a result of changing congestion levels in the network through, say, increasing the buffer in the router or by means of changing the capacity of the bottleneck link). For a related study that focuses on the buffer occupancy distributions resulting from a set of comparable open loop/closed loop simulations, we refer to [1].

4.1 Changing the bottleneck link capacity: Open loop vs. closed loop



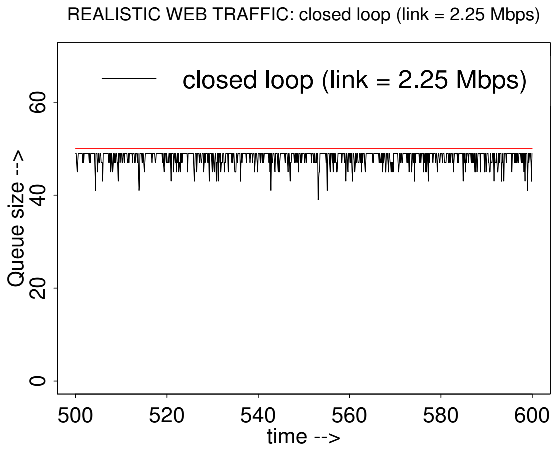


Figure 9: Open loop vs. closed loop (changing the bottleneck link capacity): Buffer occupancy processes from a *ns-2* simulation, with a 1.5 Mbps link A bandwidth, and corresponding open loop simulation assuming a link A bandwidth of 2.25 Mbps (top plot); same *ns-2* simulation, but with a 2.25 Mbps link A bandwidth (bottom plot). The horizontal reference lines indicate the maximum buffer size of 50.

To illustrate how open loop or trace-driven simulation results can give rise to misleading or wrong engineering recommendations, we consider in the following the problem of sizing the bottleneck link capacity under certain Quality-of-Service (QoS) constraints. To this end, a commonly used method in traditional performance evaluation of finite buffer queueing systems is to consider an actual packet-level traffic trace, collected from some link in, say, the Internet, and use it as input to the queueing system under consideration for a number of open loop or trace-driven simulations. For example, since changing the (constant) service or drain rate of the queue is equivalent to changing the bandwidth of the output link of the queue (which is assumed to be the bottleneck link), it has become common practice to study this way, for example, the average queue length or the (long-term) packet loss probability as a function of the bottleneck link bandwidth. Moreover, it is commonly argued that the findings from such a study apply directly to the traffic engineering problem of sizing link capacities in the presence of certain QoS requirements (i.e., small packet loss probability or small average queuelength).

In the following, we consider the network configuration depicted in Figure 1 and focus on the node 3 buffer receiving packets from link B. The maximum size of this buffer is 50, the bottleneck link A has capacity 1.5 Mbps, and we assume that the clients generate traffic according to our Web workload model (see Section 3). Next we collect the packet-level traffic trace that arrives at the buffer in question and that gives rise to a buffer occupancy processes that is depicted in the top plot in Figure 9 (solid line). Using this very trace to run a trace-driven simulation of the node 3 queueing system where now the bottleneck bandwidth has been increased to 2.25 Mbps results in the queue size process shown in the top plot of that same figure as a dotted line. Not surprisingly, the higher service rate of the queue ensures that the arriving traffic is processed faster. In fact, the resulting plot shows that the buffer tends to contain less than 20 or so packets–way below the maximum buffer size of 50–and that in contrast to the 1.5 Mbps bottleneck scenario, none of the connections alive during the depicted time interval experience any packet loss. Thus, based on these simulation results, a sound engineering recommendation would be to increase the bottleneck link capacity to reduce the average queue size as well as to guarantee essentially zero packet loss probability.

To compare, let us consider the identical *ns-2* simulation setting as above (i.e., same network configuration, same workload model) but where the bottleneck link A has now a capacity of 2.25 Mbps instead of the 1.5 Mbps considered earlier. Running this *ns-2* simulation experiment results in a node 3 buffer occupancy process that is shown in the bottom plot of Figure 9. Note that in contrast to the open loop simulation where the traffic arriving at the queue is the same irrespective of the assumed capacity of the output link, the feedback control mechanisms of TCP guarantees that a given source "learns" about the available (in this case, less stringent) bottleneck bandwidth and will adjust its sending rate accordingly. As a result, despite assuming an identical workload model at the application layer, the packet-level traffic trace arriving at the node 3 queue and generated in the closed loop environment of our *ns-2* simulation setting (with the 2.25 Mbps bottleneck speed) can be expected to be very different from that used to run the corresponding open loop simulation experiment. In fact, by constantly "probing" for their available share of the 2.25 Mbps bottleneck link bandwidth, the different competing TCP connections are not only able to send packets in general at a higher rate than in the 1.5 Mbps case, but the packets-within-connection dynamics is also likely to be different due to the different packet loss patterns exhibited by the various connections in the 1.5 Mbps vs. the 2.25 Mbps bottleneck link scenarios. In summary, the *ns-2* simulation experiment illustrates that assuming a fixed arrival stream of packets at a queue is generally ill-suited for performance modeling of closed loop systems, and that TCP is very successful at utilizing available resources (e.g., buffer and bandwidth)–at the cost of incurring potentially high loss rates, though. Clearly, based on these open loop/closed loop simulation experiments, what looked like a sound engineering recommendation in view of the open loop simulation results has turned into a potentially misleading and useless finding when accounting for more networking reality–in this case, TCP. In particular, using such open loop/closed loop simulation experiments, it can be easily demonstrated that depending on the networking conditions under which the traffic trace used in running the trace-driven simulations was gathered, the estimated performance (e.g., expressed in terms of packet loss probability) can be either overly optimistic or way too pessimistic.

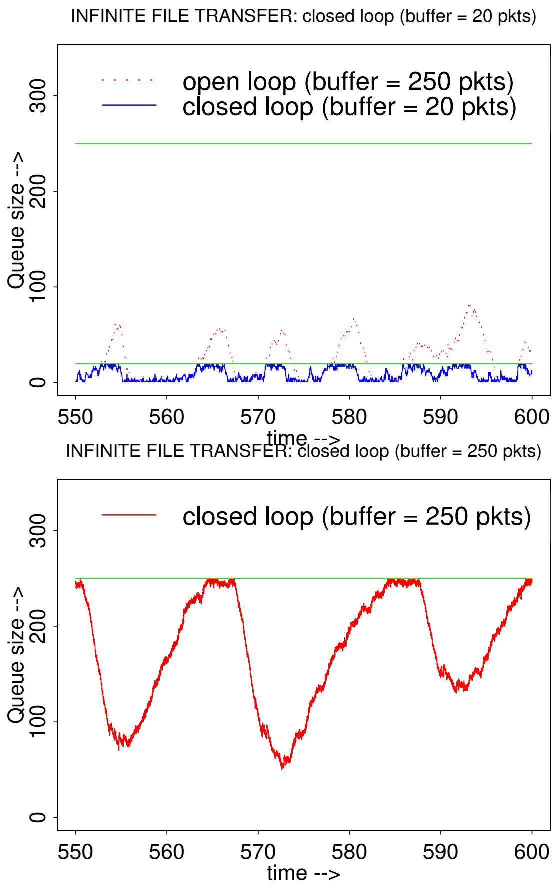


Figure 10: Open loop vs. closed loop (changing the maximum buffer size): Buffer occupancy processes from a *ns-2* simulation, with a maximum node 3 buffer size of 20 (lower reference line), and corresponding open loop simulation assuming a maximum node 3 buffer size of 250 indicated by the upper reference line (top plot); same *ns-2* simulation, but with a maximum node 3 buffer size of 250 (bottom plot).

4.2 Changing the maximum buffer size: Open loop vs. closed loop

To show that the open loop/closed loop issue is relevant irrespective of the assumed workload model, we consider again the network configuration depicted in Figure 1, focus again on the node 3 buffer receiving packets from link B (the bottleneck link C has capacity 1.5 Mbps), but assume infinite sources. Performing identical experiments as in Section 4.1 but changing the maximum buffer size instead of the bottleneck link bandwidth results in the three buffer occupancy processes depicted in Figure 10: The solid curve in the top plot results from an *ns-2* simulation run, where the node 3 buffer has a maximum size of 20; the dotted line in the top plot results from performing a trace-driven simulation of the node 3 queue in question with a maximum buffer size of 250 and with the packet-level traffic trace that was collected from the afore-mentioned *ns-2* simulation (i.e., the traffic arriving at the node 3 buffer of size 20); the bottom plot was obtained running the same *ns-2* simulation but with a maximum node 3 buffer size that has been increased from 20 to 250.

As expected and discussed in Section 3, the periodic fluctuations due to the infinite source assumption at the application layer dominates all three curves. However, the impact of the TCP feedback control that is explicitly accounted for in the *ns-2* simulations but completely ignored in the trace-driven simulation is as clear and qualitatively the same as in the experiments discussed in the previous section. Again, TCP fully exploits and utilizes (to the extent possible when dealing with infinite sources) the increased node 3 buffer capacity but does so at the cost of incurring packet losses at times of congestion; that is, when the overall rate at which the packets arrive at node 3 exceeds the output rate for too long of a period. Similar conclusions as in the previous section apply: While at first sight, the open loop simulation results suggest a clear advantage of increasing the maximum buffer size from 20 to 250, their closed loop counterparts fully discount this observation as an illusion due to relying on simplified models that miss out on aspects of traffic (e.g., the TCP feedback mechanism) that turn out to be of crucial importance for understanding network performance.

4.3 Changing the workload model: Open loop vs. closed loop

Finally, we comment on another commonly-used approach for inferring buffer dynamics in an Internet-like setting from open loop trace-driven simulations, where in addition, the workload model comes into play. It is common engineering practice to use the complementary probability distribution of the buffer occupancy in an infinite buffer queue as an accurate substitute for the loss probability in a finite buffer system. To check the validity of this practice in our simple networking setting, we run a number of identical *ns-2* simulations, except that we considered different maximum buffer occupancies at node 3, namely $10, 20, 30, 50, 250$ and 1000, and obtained the actual loss probabilities as a function of the buffer size at the node 3 queue. As our infinite buffer system, we take the simulation with maximum buffer size of 1000 and infer from it the complementary probability distribution function that the buffer exceeds a certain value x. The results are depicted in Figure 11 (the solid lines correspond to the infinite buffer approximation, while the crosses indicate the actual loss probabilities), where the top plot is for the infinite source case and the plot at the bottom is based on the Web sources. For the infinite sources and a maximum buffer size of 1000, the queue length process resulting from the *ns-2* simulation run (i.e., closed loop) turns out to tightly fluctuate around 725. In fact, the maximum buffer capacity is never exceeded, resulting in no losses and

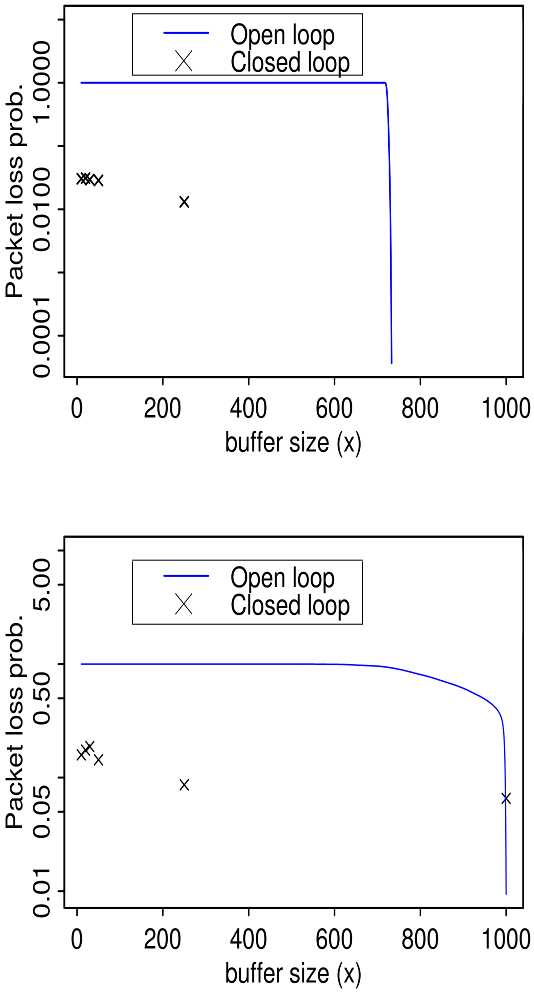


Figure 11: Open loop vs. closed loop (changing the workload model): Packet loss probability vs./ buffer size, for the infinite source model (top plot) and the Web source model (bottom plot).

implying that in this scenario, the sources are bandwidth limited on their access links. In contrast, the *ns-2* simulation for the 1000 buffer size case with Web sources does occasionally fill the whole buffer resulting in significant losses of more than 5%. On the other hand, note that the infinite buffer approximations are by definition open loop based and predict, for example, almost 100% packet losses for the 50 buffer case, even though the actual packet losses are below 15% for Web sources and below 3% for the infinite sources. Overall, Figure 11 illustrates that the infinite buffer approximation can lead to extremely conservative performance prediction, making this open loop-based approach to inferring aspects of a closed-loop system essentially useless.

5. CONCLUSION

Even though the networking configuration considered in our simulation experiments is admittedly unrealistic and over-simplified, experimental evidence presented in this paper exemplifies the risk associated with conventional analysis or simulation of large-scale internetworks such as the Internet. As cogently discussed in [11, p. 2], this risk concerns the wide-spread tendency to rely on and *"use models simplified to the point where key facets of Internet behavior have been lost, in which case the ensuing results could be useless (though they may not appear to be so!)."* While simple toy networks, infinite source models, and open loop systems can provide deep insight into and physical understanding of the performance of real networks, we believe that their credibility should be substantially enhanced by demonstrating that after accounting for realistic network, source model, and feedback behavior, the insight and understanding they provide (i) remain essentially unchanged, or (ii) may require substantial modifications, or (iii) are no longer applicable. For example, our experimental results clearly indicate that conventional analysis or simulation cannot be expected to automatically apply in Internet-like settings where a major portion of the traffic is generated by Web users (and hence gives rise to high variability TCP connections) and uses TCP (and hence is inherently closed loop in nature).

Looking ahead, it will be interesting to see whether or not some of the generic differences observed in our abstract setting will remain valid for more realistic network configurations. Another less obvious shortcoming of our experiments presented in this paper is that we completely ignore the potential of feedback from the network all the way back to the application layer; that is, the congestion state of the network may have a direct impact on our Web-user source model because it may directly influence the Web-browsing behavior of individual users. While there exists mainly anecdotal evidence for the presence of such types of feedback behavior (e.g., Internet "storms" [13]), we have seen little empirical evidence for the widespread existence of such feedback in our analysis of a wide variety of Internet traffic measurements. Nevertheless, the potential pitfalls associated with assuming an open loop characterization at the source level should be kept in mind and may require revamping the current approach to source modeling, depending on how the Internet develops in the near future. Other aspects not considered in our experimental studies concern incorporating TCP features such as SACK (selected ack) or delayed acks [15]; dealing with the problem of two-way or cross traffic (e.g., see [23, 5]); allowing for more realistic networking topologies; and a more comprehensive analysis of the buffer occupancy in routers running RED. Part of our ongoing efforts to understand the dynamics of TCP traffic in a realistic networking setting deals with some of these aspects and how they impact our current understanding, and will be published elsewhere.

Acknowledgments

We would like to acknowledge Polly Huang for her help with the *ns-2*-simulations and Sally Floyd for valuable comments and providing references to related studies. We are also grateful to the reviewers for constructive criticism and helpful suggestions for sharpening the focus of the paper and for improving the overall presentation of the material. An earlier version of this paper was presented in an invited session at the 37th Annual Allerton Conference on Communication, Control, and Computing, Allerton House, Monticello, IL, September 22-24, 1999, [16].

6. REFERENCES

[1] A. Arvidsson and P. Karlsson. On traffic models for TCP/IP. In: *Teletraffic Engineering in a Competitive World*, Proc. ITC-16, Edinburgh, UK, P. Key and D. Smith (Eds.), North-Holland, Amsterdam, pp. 457–466, 1999.

[2] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. McCanne, R. Rejaie, P. Sharma, S. Shenker, K. Varadhan, H. Yu, Y. Xu, and D. Zappala. Virtual InterNetwork Testbed: Status and research agenda. Technical Report 98-678, University of Southern California, July 1998.

[3] P. Barford and M. E. Crovella. Generating representative web workloads for network and server performance evaluation. In *Proceedings of Performance'98/ACM Sigmetrics'98*, pages 151–160, 1998.

[4] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic - evidence and possible causes. In *Proceedings of ACM Sigmetrics'96*, pages 160–169, 1996.

[5] A. Feldmann, A. C. Gilbert, P. Huang, and W. Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. In *Proc. of the ACM/SIGCOMM'99*, pages 301–313, Boston, MA, 1999.

[6] A. Feldmann, A. C. Gilbert, and W. Willinger. Data networks as cascades: Investigating the multifractal nature of Internet WAN traffic. In *Proc. of the ACM/SIGCOMM'98*, pages 25–38, Vancouver, B.C., 1998.

[7] A. Feldmann, A. C. Gilbert, W. Willinger, and T. G. Kurtz. The changing nature of network traffic: Scaling phenomena. *Computer Communication Review*, 28(2):5–29, 1998.

[8] S. Floyd and V. Jacobson. On traffic phase effects in packet-switched gateways. *Internetworking: Research and Experience*, 3(1):115–156, 1992.

[9] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[10] S. Floyd and V. Jacobson. The Synchronization of Periodic Routing Messages. In *Proc. of the ACM/SIGCOMM'98*, pages 33–44, 1993.

[11] S. Floyd and V. Paxson. Why we don't know how to simulate the Internet. Preprint, 1999. (An earlier version of this paper appeared in: *Proceedings of the 1997 Winter Simulation Conference*, Atlanta, GA, 1997.)

[12] C. Hollot, V. Misra, D. Towsley, and W. Gong. A control-theoretic analysis of RED. *Proc. of the IEEE INFOCOM'01*, 2001 (to appear).

[13] B. A. Huberman and R. M. Lukose. Social dilemmas and Internet congestion. *Science*, 277:535–537, 1997.

[14] V. Jacobson. Congestion Avoidance and Control. In *Proc. of the ACM/SIGCOMM'88*, pages 314–329, 1988.

[15] V. Jacobson, R. Braden, and D. Berman. TCP extensions for high performance, May 1992. Request for Comments 1323.

[16] Y. Joo, V. Ribeiro, A. Feldmann, A.C. Gilbert, and W. Willinger. On the impact of variability on the buffer dynamics in IP networks. In *Proc. of the 37th Annual Allerton Conference on Communication, Control, and Computing, Allerton House, Monticello, IL, September 22-24, 1999, Allerton, IL*, 1999.

[17] V. Paxson and S. Floyd. Wide area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3:226–244, 1995.

[18] S. Shenker, L. Zhang, and D. Clark. Some Observations on the Dynamics of a Congestion Control Algorithm. *ACM Computer Communication Review*, 20(4):30–39, October, 1990.

[19] W.R. Stevens. *TCP/IP Illustrated Volume 1*. Addison-Wesley, 1994.

[20] W. Willinger, V. Paxson, and M. S. Taqqu. Self-similarity and heavy tails: Structural modeling of network Traffic. in: *A Practical Guide to Heavy Tails: Statistical Techniques for Analyzing Heavy Tailed Distributions*, R. Adler, R. Feldman and M.S. Taqqu (Eds.), pp. 27–53, Birkhauser Verlag, Boston, MA, 1998.

[21] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5:71–86, 1997.

[22] L. Zhang and D. Clark. Oscillating Behavior of Network Traffic: A Case Study Simulation. *Internetworking: Research and Experience*, 1(2):101–112, 1990.

[23] L. Zhang, S. Shenker, and D. Clark. Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic. In *Proc. of the ACM/SIGCOMM'91*, pages 133–147, 1991.