Simulation Perspectives on Link Buffer Sizing

Sergey Gorinsky
Anshul Kantawala
Jonathan Turner
Applied Research Laboratory
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis
MO 63130-4899, USA
gorinsky@arl.wustl.edu

The problem of determining the optimal buffer size for an Internet link has recently attracted the strong revived attention of networking scientists. While consonant in dissenting from the conventional wisdom to set the buffer size to the product of the link bitrate and round-trip propagation delay of served connections, the new studies often propose alternative guidelines that starkly contradict each other. In this paper, we review the problem of link buffer sizing from two simulation perspectives. First, we use the packet-level simulator ns-2, which is currently the predominant tool for evaluation of network designs, to explore the contradictions between the buffer sizing guidelines. We attribute the contradictions to differences in assumptions and goals of the earlier studies and present our own argument for using small constant buffer sizes. Second, we reflect on the suitability of ns-2 itself for studying the problem of link buffer sizing. We the identify ns-2 deficiencies that undermine the trustworthiness of results provided by this prevailing simulation tool. We also suggest directions and discuss challenges for designing a simulation methodology capable of providing undisputed answers on link buffer sizing.

Keywords: network congestion control, link buffer sizing, ns-2 simulations

1. Introduction

In this paper, we investigate how much buffer an Internet router should allocate for its output link. This problem of link buffer sizing has recently become a topic of intensive and sometimes heated research discussion. To explore and explain dramatic contradictions between proposed guidelines for setting the buffer size, we conduct experiments in ns-2 [1], a predominant simulation tool in the networking research community. One important conclusion is that the contradictions are due to made assumptions and pursued objectives of the prior studies. This finding implies that a consensus on the right answer for link buffer sizing necessitates a general agreement on what constitutes the right question. Our opinion is that the problem has to be formulated in harmony with the overall philosophy of the Internet design.

SIMULATION, Vol. 83, Issue 3, March 2007 245–257

DOI: 10.1177/0037549707081179

The tremendous growth of the Internet is often attributed to its architectural minimalism. The Internet Protocol (IP) merely defines a format of datagrams [2]. End systems communicate by sending datagrams over a series of links and routers. IP expects from routers nothing but best-effort forwarding of datagrams to appropriate output links. Other than for enabling the forwarding function, no other signaling between Internet routers is required.

While the minimalism of signaling requirements facilitates rapid deployment of IP networks, the Internet suffers from congestion when an output link of a router receives datagrams at a rate exceeding the link bitrate. If overload persists, the router buffers and eventually discards datagrams. To control congestion, end systems are expected to curb their transmission upon inferring congestion from implicit signs such as datagram loss. Many applications rely on the Transmission Control Protocol (TCP) [3] for congestion control [4, 5]. Other applications communicate over the User Datagram Protocol (UDP) that does not provide a congestion control service [6] and therefore have to employ their own mechanisms for reducing transmission upon a link overload.

Internet congestion control rests on the principle of perpetual end-to-end probing for available network capacity. Even when the set of connections sharing a bottleneck link does not change, their total load on the link does not converge to a single value but oscillates within a range [7]. If the link buffer is too large, datagrams experience unnecessarily long queuing delay. If the buffer is too small, the load oscillations lead to incomplete utilization of the link.

Link buffer sizing is traditionally formulated as a problem of utilizing the bottleneck link fully without excessive queueing. Solving the problem depends on the buffer management and link scheduling policies of the router. Throughout this paper, we focus on FIFO (First In, First Out) Droptail links which discard datagrams only upon buffer overflow and forward queued datagrams in the order of their arrival.

Another factor in link buffer sizing is the congestion control exercised by end systems. The conventional formulation of the problem assumes TCP congestion control. Simple analysis of queuing at a link that serves a single TCP connection in the congestion–avoidance mode yields the traditional guideline of setting the buffer size to the product of the link bitrate and round-trip propagation delay of the connection [8]. More recent studies argue that as the number of TCP connections increases, the total range of load oscillations and hence the optimal buffer size shrink [9, 10]. An alternative line of reasoning maintains that the optimal buffer size should grow proportionally to the number of TCP connections to avoid high datagram loss that hampers TCP performance [11–13].

In this paper, we use simulations to understand why the aforementioned guidelines, which are all geared to accommodate TCP congestion control, suggest such starkly contradictory amendments for scenarios with multiple TCP connections. We determine that the guidelines are derived for legitimate but different sets of network configurations. We proceed by reconciling the contradictions with a unifying rule that works over a comprehensive scope of parameter settings.

We also argue that the above traditional focus of link buffer sizing on optimization for TCP is misguided. The problem should be reformulated to be harmonious with the Internet design philosophy of using minimal signaling between routers and accommodating various applications, including those that communicate over UDP. As in the earlier version of this work [14] and other recent investigations [10, 15, 16], we assert that the best trade-off in link buffer sizing is one that offers negligible queuing delay to all Internet applications. Our proposal calls for small constant buffers. Although small buffers underutilize the bottleneck link capacity and cause high datagram loss in many realistic scenarios involving TCP connections, we agree with Gu et al. [17] that the suboptimal performance should be rectified not by longer buffering at routers but through improvement of end-to-end congestion control. We discuss smoother transmission and other promising techniques for effective congestion control with small link buffers.

Simulation is not only a tool but also a subject of our studies. The lessons learned include a common-sensible insight that simulations should examine link buffer sizing over the whole range of realistic parameter settings. However, even our own investigation does not follow this principle due to the inability to complete ns-2 simulations in scenarios in which tens of thousands of concurrent connections share a high-speed network. While the scalability limitations of ns-2 call for alternative simulation techniques, this desire conflicts with the realization that trustworthy answers on link buffer sizing require more detailed simulations than ns-2 provides. In particular, our discussions with prominent researchers raise concerns that short timescale modeling of both routers and end systems in ns-2 is insufficiently precise in order to offer undisputed conclusions on loss processes at routers and thus on optimal sizes of link buffers. We discuss avenues for addressing the above challenges in an appropriate simulation methodology for the problem of link buffer sizing.

The rest of the paper is structured as follows. Section 2 describes earlier proposed guidelines for link buffer sizing. Section 3 uses simulations to explain and reconcile contradictions between the guidelines. Section 4 discusses challenges in making the simulations trustworthy as well as directions for addressing these challenges. Section 5 argues that the problem of link buffer sizing has to be reformulated to be in harmony with the overall Internet design philosophy. Section 6 proposes a method to solve the newly formulated problem by using small constant buffers. Section 7 discusses congestion control techniques that are capable of providing such applications as long file transfers with high performance despite small link buffers. Section 8 finishes the paper with a summary.

2. Discordant Rules for Link Buffer Sizing

Bitrate-delay product is the traditional rule of thumb that prescribes setting the link buffer size to the product of the link bitrate and round-trip propagation delay. Whereas the guideline is often attributed to Villamizar and Song [18], the rule was known earlier; for example, Jacobson described the rule clearly in his posting to the end2end-interest mailing list [8]. The bitrate-delay-product guideline is easily derivable from a model in which a FIFO Droptail link serves a single TCP connection. The model assumes that the connection operates with a repetitive pattern where the sender halves its load on the network upon overflowing the buffer and then increases the load additively until the next overflow occurs [7]. The assumed Additive-Increase Multiplicative-Decrease (AIMD) pattern approximates well the congestion-avoidance mode in such versions of TCP as Reno [8] and NewReno [19]. After an overflow of the link buffer, the sender decreases its load on the network from $2B$ to B. If the product of the link

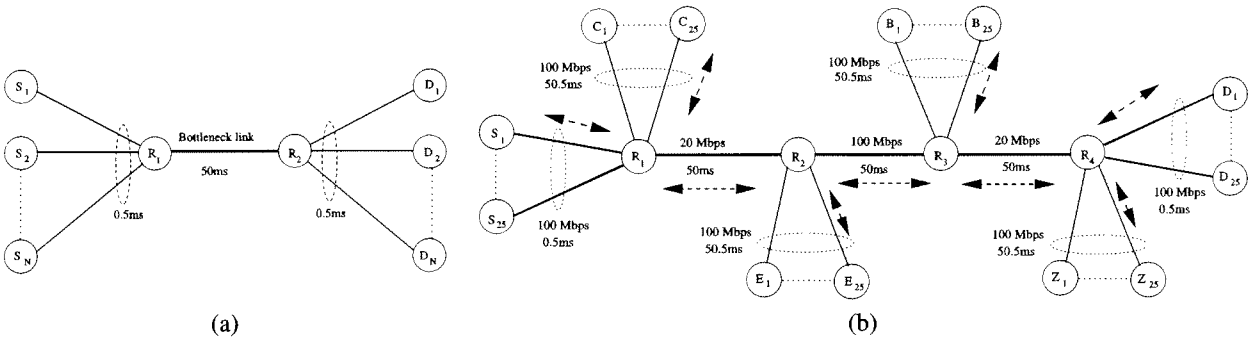

Figure 1. Network topologies in our experiments: (a) Single-bottleneck topology; (b) Multiple-bottleneck topology.

bitrate and round-trip propagation delay of the connection equals B, then the decrease drains the buffer completely without under-utilizing the link. Since the buffer size in this setting also equals B, the analysis concludes that the bitrate-delay product is the optimal size for the link buffer.

More recent analyses enhance the above simple model by including scenarios where more than one TCP connection traverse the bottleneck link. These analyses provide mutually contradictory guidelines, to which we refer below as over-square-root and connection-proportional allocation.

Over-square-root is an interesting generalization of the traditional rule; its proposal has triggered a new wave of research on link buffer sizing. The guideline prescribes setting the buffer size to the bitrate-delay product divided by $\sqrt{n}$ where n is the number of TCP connections [9]. The refinement is backed by experimental evidence that some connections lose no datagrams during an overflow of the buffer. Due to the consequent asynchrony of individual load reductions, the amplitude of total load oscillations is smaller for a larger number of connections [9, 20]. Therefore, the over-square-root guideline proposes reducing the link buffer size as the number of TCP connections grows.

Connection-proportional allocation is a family of radically different guidelines suggesting that the optimal buffer size increases proportionally to the number of TCP connections [11–13]. Such guidelines stem from an observation that TCP connections suffer from high datagram loss and frequent retransmission timeouts unless the link buffer is large enough to allow each connection at least few unacknowledged data segments. The connection-proportional allocation is not necessarily in conflict with the conventional rule of thumb. For example, Dhamdhere et al. [11] propose a buffer sizing formula that approximates the bitrate-delay product closely when the number of TCP connections is small.

3. Simulation Insights into the Contradictions

The guidelines of connection-proportional allocation and over-square-root agree on the inadequacy of the traditional rule in scenarios with numerous TCP connections but are in a stark conflict on whether the buffer size should be increased or decreased as the number of connections grows. In this section, we use simulations to understand reasons behind the contradictions.

3.1 Simulation Setup and Results

First, we perform ns-2 simulations in a common single-bottleneck network topology presented in Figure 1a. The sending and receiving ends of unicast applications are located at end systems S_i and D_i respectively. Since the value of i varies from 1 to N, the network contains $2N$ end systems. All the applications communicate data via datagrams of size 1500 bytes. All the links employ FIFO Droptail buffering. The link from router R_1 to router R_2 is a bottleneck for each of the applications. This link has a propagation delay of 50 ms. Each of the other links has a propagation delay of 0.5 ms and a bitrate that is twice higher than the bottleneck bitrate. Every experiment lasts 200 s.

During an initial series of our simulations, each of the N applications transfers a large file over TCP NewReno. We conduct the experiments for three settings of the bottleneck link bitrate where the bitrate equals 100, 10, and 1 Mbps, respectively. For file transfers, full utilization of the bottleneck link is not a goal in itself. What is important for this application is the file delivery time. Hence, a file transfer views the link buffer size as optimal if it maximizes *end-to-end goodput* defined as the ratio of the file size to the file delivery time. In addition to end-to-end goodput, we also track loss rates and round-trip times. All the measurements are collected over the entire duration of each experiment.

Our results appear in Figures 2 and 3. For the bottleneck bitrate of 100 Mbps, the original rule approximates the optimal buffer size precisely. Figure 2a offers no solid justification for making the buffer size either smaller or larger than the bitrate-delay product. For the bottleneck bitrate of 10 Mbps, the graphs exhibit a different trend after the number of TCP NewReno connections exceeds 50.

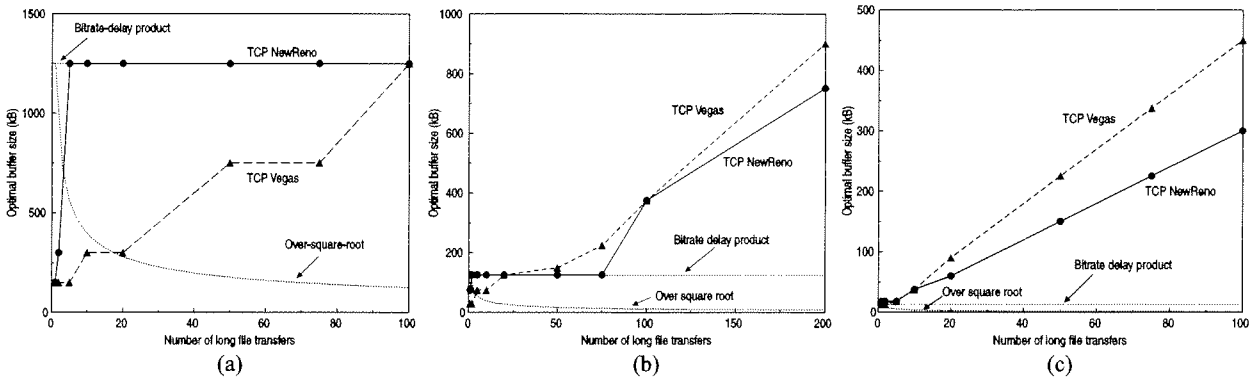

Figure 2. Optimal buffer size for file transfers over TCP: (a) 100 Mbps bottleneck link; (b) 10 Mbps bottleneck link; (c) 1 Mbps bottleneck link.

Instead of remaining constant at the bitrate-delay product, the optimal buffer size starts growing. The growth is close to linear and therefore consistent with the connection-proportional allocation. The bottleneck bitrate of 1 Mbps provides even stronger support for this guideline: Figure 2c shows that the optimal buffer size is approximately proportional to the number of TCP connections over the whole explored range from 1 to 100 connections. Neither of the ns-2 simulations justifies reducing the buffer size as the number of connections grows. From the presented simulation perspective, the connection-proportional allocation obviously seems as a better guideline than over-square-root.

3.2 Analysis of the Results

To start discussing the simulation results, let us focus on the bitrate-delay product and connection-proportional allocation. Figure 2 indicates that applicability of these guidelines depends on the ratio of the bitrate-delay product to the number of TCP NewReno connections. If the ratio is high, the bitrate-delay product is a precise approximation for the optimal buffer size. If the ratio is low, the optimal buffer size is consistent with the connection-proportional allocation. What makes the ratio such an important parameter? Our verification confirms correctness of the logical derivations that lead to establishment of both guidelines. The contradictory recommendations of the guidelines are due to differences in made assumptions and pursued goals. The original rule strives to utilize the bottleneck link fully and assumes that the single TCP connection operates uninterruptedly in the congestion-avoidance mode. However, if the bottleneck link buffer is not large enough to allow the connection at least few unacknowledged data segments, the assumption of the congestion-avoidance mode is not tenable due to high datagram loss and consequently frequent retransmission timeouts. On the other hand, the connection-proportional allocation pursues an objective of keeping each TCP connection in the congestion-avoidance mode but does not concern itself with the bottleneck link utilization. Therefore, when the ratio of the bitrate-delay product to the number of connections is high, the connection-proportional allocation fails to optimize the end-to-end goodput of the TCP connections due to underutilization of the bottleneck link.

By examining the wide range of values for the ratio of the bitrate-delay product to the number of connections, our simulations explain the contradictions between the rules of bitrate-delay product and connection-proportional allocation. Moreover, the simulations pave the way for reconciling the contradictions with a unified guideline: to keep each TCP connection in the congestion-avoidance mode and utilize the bottleneck link fully, *set the link buffer size to the maximum of the bitrate-delay product and connection-proportional allocation.* It is interesting that our reconciling rule exhibits a striking structural similarity with the guideline proposed by Dhamdhere et al. [11].

We now turn our attention to over-square-root. Figure 2 shows clearly that the guideline fails in many realistic scenarios. In particular, when 100 TCP connections share the 100 Mbps bottleneck link, the optimal buffer size equals the bitrate-delay product; setting the buffer size in accordance with the over-square-root guideline to 10% of the bitrate-delay product provides the file transfers with smaller goodput due to lower utilization of the bottleneck link. One explanation for the lack of any evidence in support of over-square-root is that our simulations explore settings different from those intended by the guideline. The original analytical justification for the over-square-root rule considers a congested high-bitrate link in the Internet core and assumes that tens of thousands of connections compete for this backbone bottleneck. Moreover, Appenzeller et al. [9] indicate that desynchronization of load reductions appears and reveals the

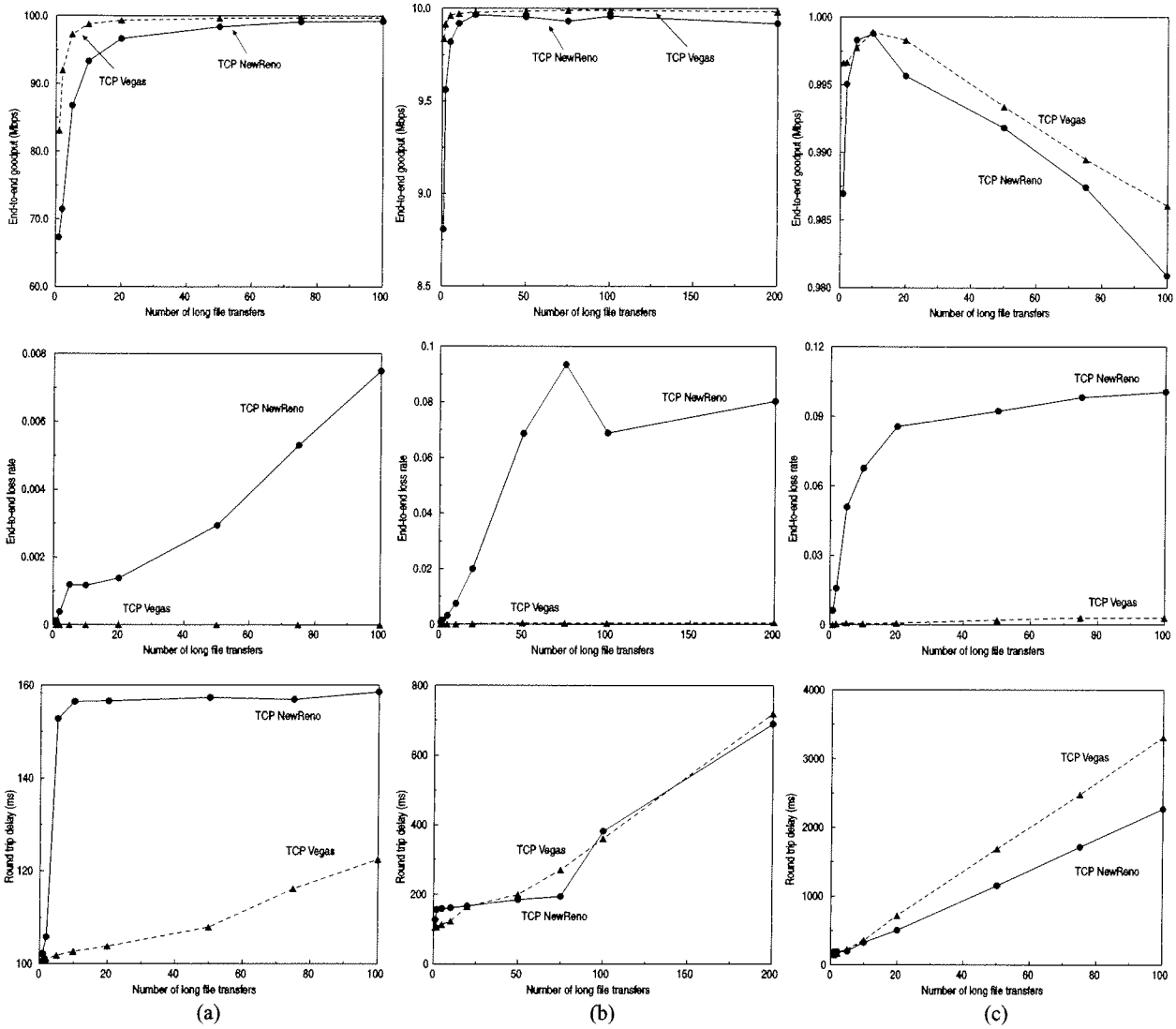

Figure 3. End-to-end goodput, loss rate, and round-trip delay when the link buffer size is optimal: (a) 100 Mbps bottleneck link; (b) 10 Mbps bottleneck link; (c) 1 Mbps bottleneck link.

$\sqrt{n}$ adjustment only when n is at least on the order of thousand connections. Due to scalability limitations of ns-2, we are unfortunately unable to repeat the simulations in high-bitrate network topologies with tens of thousands of concurrent connections. However, if such large-scale simulations did show desynchronization of load reductions in high-multiplexing settings, it would be possible to amend our reconciling guideline accordingly.

4. Trustworthiness of Simulation Insights

The discussion in the end of Section 3.2 prompts us to shift the focus of the paper temporarily from link buffer sizing onto the simulation methodology for studying this problem. One lesson learned is a rather common sense realization that simulations should strive to examine buffer sizing guidelines over the entire scope of realistic parameter settings. Although we agree with the simulation tip by Wischik and McKeown [10] that it is convenient to study the effects of multiplexing (i.e., the number of TCP connections) and of the individual window size (i.e., the ratio of the bitrate-delay product to the number of TCP connections) separately, the separation should not come with ignoring either of the dependencies or exploring them within a limited scope. In particular, our experience demonstrates the great importance of conducting

simulations over a wide range of values for the ratio of the bitrate-delay product to the number of connections.

Dependability of evidence supplied by ns-2 simulations has emerged as a major issue in our consultations with Appenzeller [pers. comm., 2005], McKeown [pers. comm., 2005], and Jeffay [pers. comm., 2005]. The lack of consensus on validity of our simulation results in Section 3 is primarily due to concerns whether ns-2 constitutes a trustworthy evaluation platform for link buffer sizing. A more specific fear is that short-timescale modeling of both routers and end systems in ns-2 is insufficiently precise to represent loss processes at routers realistically; from such standpoint, ns-2 seems to synchronize the loss processes and inflate the buffer requirements significantly whereas real networks exhibit little or no evidence of the loss synchronization even with as few as 50 TCP connections [Appenzeller, G., pers. comm., 2005; McKeown, N., pers. comm., 2005].

Hence, it becomes clear that converging to a universally accepted guideline for link buffer sizing necessitates a general agreement on appropriate simulation methods. Since the inherent randomness of network delays is known to affect the loss synchronization [21], more modeling research is needed on probabilistic characterization of delays experienced by datagrams at routers and end systems. As suggested in earlier studies [10], the closely related factor of traffic burstiness also requires a careful consideration.

Incorporation of more detailed modeling is a formidable challenge because of the conflicting need in improved scalability of simulations. In accordance with the approach advocated by Floyd and Kohler [22], we argue for problem-specific modeling: instead of relying on a universal network simulator such as ns-2, simulations of link buffer sizing should be conducted in a specialized model that represents important properties (e.g., of delay dynamics) precisely but makes simplifying assumptions about less relevant aspects (e.g., routing).

The last, but not least, issue with an appropriate simulation methodology for link buffer sizing is its validation. Laboratory facilities that use personal computers to emulate a wide-area network [12] are not immune against the above concerns about trustworthiness of acquired results. For instance, dummynet [23] that imitates link propagation delays in many such facilities makes traffic inadvertently more bursty and thereby contributes to exaggeration of the buffer requirements [Appenzeller, G., pers. comm., 2005]. Therefore, experiments in real networks appear to be the most promising avenue for reaching a consensus on validity of simulation techniques for link buffer sizing. Conducting these experiments is a challenge in itself because they necessitate collaboration with operators of high-bitrate networks in the Internet core.

5. Reformulation of Link Buffer Sizing

After discussing challenges and directions for making the simulation methodology trustworthy in Section 4, we now return to the question what constitutes the right guideline for link buffer sizing. We argue that the original focus on optimization for TCP is misguided. The problem should be reformulated to be in harmony with the Internet design philosophy of using minimal signaling between routers and accommodating various applications.

Flaws in the current approach to the problem reveal themselves already in Figure 2. The figure shows that the optimal buffer size for Vegas [24] version of TCP does not depend on the bitrate-delay product but remains approximately proportional to the number of connections. With TCP Vegas, the optimal buffer size is different due to a different algorithm employed for load adjustments: since Vegas relies on Additive-Increase Additive-Decrease (AIAD) in the congestion-avoidance mode, the total load oscillations are proportional to the number of connections. The failure of our NewReno guideline to identify the optimal buffer size for Vegas prompts us to spell out the traditionally implied formulation of the buffer sizing problem as follows: *find the buffer size that optimizes performance of long file transfers over a particular TCP version.*

Our position is that the traditional problem formulation is inadequate. Buffer sizing should accommodate not only a specific TCP version or application class but also other types of Internet traffic. Besides, a useful formulation of the problem should explicitly require that a proposed guideline is implementable within the Internet architecture. These two issues are explored in detail in Sections 5.1 and 5.2 respectively.

5.1 Buffer Sizing for Different Applications

Although long file transfers over TCP are common in the Internet, the Internet also serves other traffic such as short file transfers over TCP and interactive streaming over UDP. Link buffer sizing impacts these other applications as well but with qualitatively different effects.

Short file transfers over TCP do not operate in the congestion-avoidance mode for long. Unlike with large files, it is not AIMD dynamics anymore but other factors that dominate optimal buffer sizing. In particular, queueing at the bottleneck link contributes heavier to file delivery time as files become smaller. According to recent studies, performance for an arbitrary number of short file transfers is optimal with small constant buffers [9, 25, 26].

Queueing delays are even more consequential for interactive streaming over UDP because interaction at round-trip times larger than few hundred milliseconds makes humans uncomfortable. Whereas such applications reject TCP due to its jittery transmission and extra delay of its reliable in-order delivery, communication over UDP comes

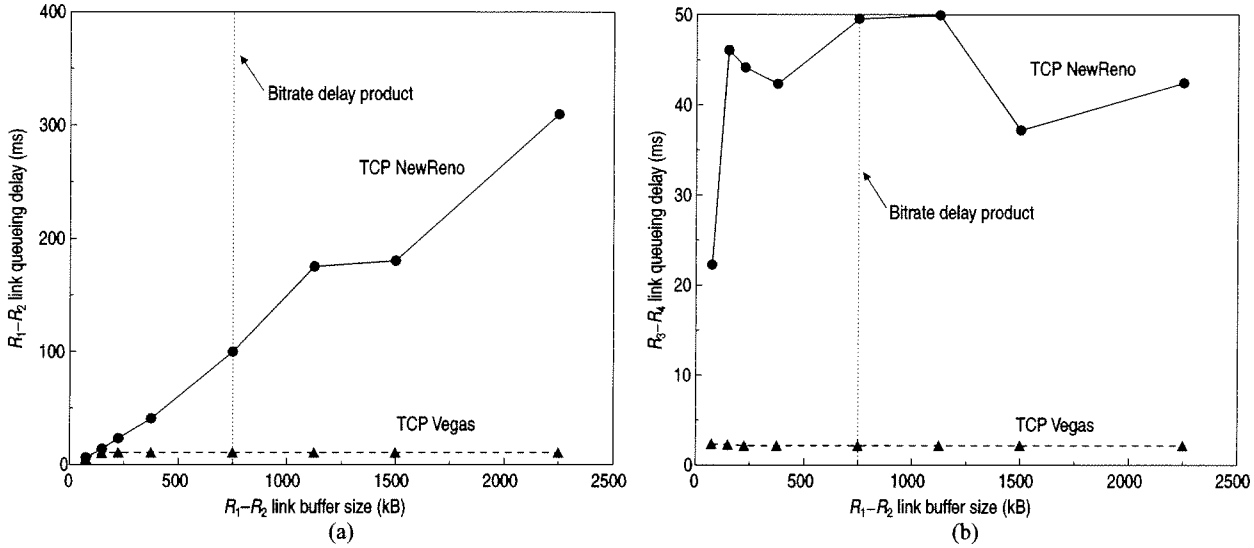

Figure 4. Link queueing delays in the multiple-bottleneck topology: (a) Link R_1–R_2; (b) Link R_3–R_4.

with responsibility of performing own congestion control. The choice of the congestion control algorithm greatly affects the optimal buffer size for the streaming application.

As Figure 3c manifests with queuing delays on the order of seconds, selecting the link buffer size to optimize performance of long file transfers can cause prolonged queueing. Such extensive queueing might be undesirable for short file transfers and interactive streaming. To quantify the impact of link buffer sizing on different types of Internet applications, we conduct additional ns-2 experiments in the multiple-bottleneck network topology depicted in Figure 1b. The topology represents a common Internet scenario where backbone links are utilized lightly, and bottlenecks are at access links. The network contains four routers R_1, R_2, R_3 and R_4 and carries balanced bidirectional traffic on each link. There are three groups of traffic: between end systems S_i and D_i, between end systems C_i and E_i, and between end systems B_i and Z_i, where i varies from 1 to 25. The following applications form each group: an interactive video application transmitting one constant-bitrate 2 Mbps stream over UDP in each direction; eight long file transfers over TCP, four in each direction; short web downloads, twenty sources in each direction. For the web downloads, every source periodically generates a 36-kB data burst, transmits it over TCP, and then goes idle for a time interval that has a duration distributed exponentially with the mean of 1 s. All the applications use data-grams of size 1500 bytes. TCP applications employ either NewReno or Vegas. Each link uses FIFO Droptail buffering. All the applications have the same round-trip propagation delay. Hence, the bitrate-delay product for a link is the same for every application using this link. Our evaluation focuses on the applications transmitting data from end systems S_i. Links R_1–R_2, R_3–R_4, R_4–R_3, and R_2–R_1 are the four bottlenecks for these applications. Once again, each experiment lasts 200 s, and we measure link loss rates, queueing delays, and utilizations over the entire duration of the experiment.

Queueing delays at links R_1–R_2 and R_3–R_4 are reported in Figure 4 for scenarios where the buffer size of link R_1–R_2 varies but the buffer sizes of all the other links are set to their bitrate-delay products. Figure 5 plots end-to-end performance metrics that are meaningful for each type of the examined applications: round-trip time for interactive video, goodput for long file transfers, and delivery time for short web downloads. For TCP NewReno, the traditional guideline of setting the buffer size for link R_1–R_2 to its bitrate-delay product results in low link loss rates and high link utilizations for both links R_1–R_2 and R_3–R_4, large end-to-end goodputs for the file transfers, and small delivery times for the web downloads. On the other hand, the bitrate-delay-product setting punishes interactive video with unacceptably large round-trip times exceeding 600 ms.

Our simulations demonstrate that choosing a large buffer to accommodate long file transfers can lethally hurt interactive streaming. Can a smaller buffer reconcile the needs of all the applications? Earlier studies reply affirmatively by showing that small buffer sizes do not disrupt file transfers substantially in terms of average goodput, even though individual goodputs can become more variable [21, 26]. To examine the sensitivity of end-to-end goodput to suboptimal buffer sizes, we repeat the experiments from Section 3.1 by setting the buffer size of the 10 Mbps bottleneck link to different fractions of the bitrate-delay product BD. As expected, Figure 6 shows that end-

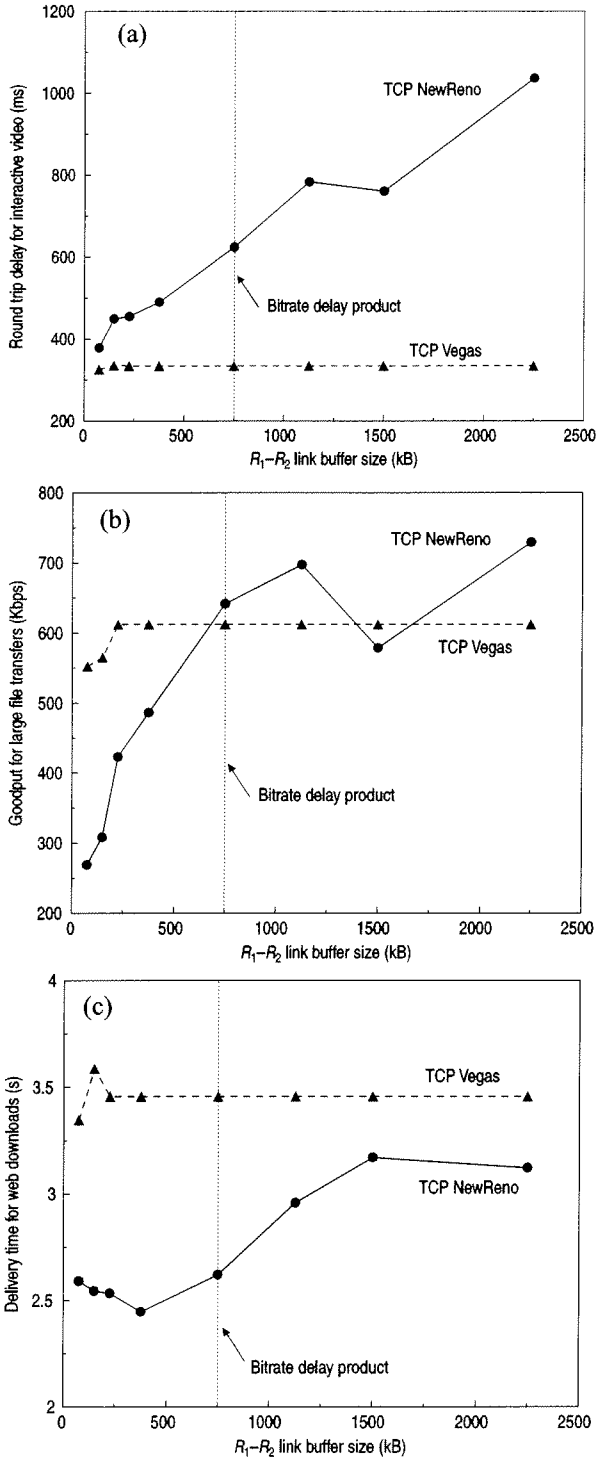

Figure 5. Performance of different applications in the multiple-bottleneck topology: (a) Round-trip time of interactive video; (b) End-to-end goodput of long file transfers; (c) Delivery time of short web downloads.

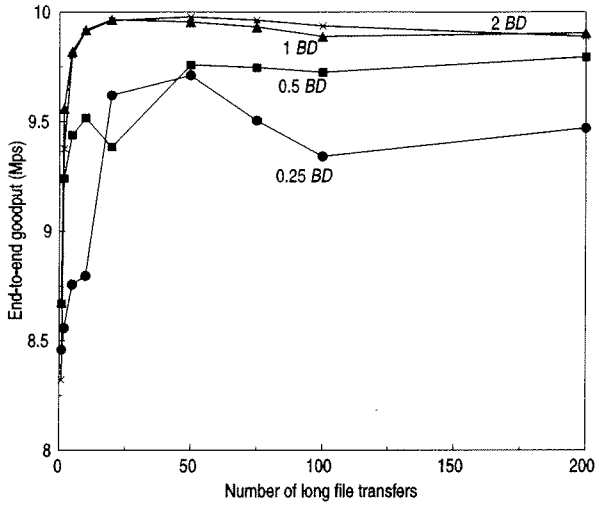

Figure 6. Impact of suboptimal buffer sizes on long file transfers over TCP NewReno.

to-end goodput decreases when the fraction reduces from 1 to 0.5 and further to 0.25. However, the extent of the decrease is not substantial and does not grow together with the number of long file transfers. Therefore, long file transfers can achieve good performance even with low buffer sizes.

5.2 Minimalism of Internet Signaling

As we conclude in Section 5.1, reconciling diverse needs of Internet applications favors small buffers. Before we describe our specific solution for link buffer sizing in Section 6, we now argue that the ability to implement a buffer sizing guideline within the Internet architecture is extremely important and thus has to be included as an explicit constraint into the problem formulation.

The overall network design philosophy plays an implicit but major role in traditional formulations of the buffer sizing problem. Due to the minimalism of signaling, the Internet relies on end systems to control congestion without enabling them to notify routers about buffer requirements of applications. In more sophisticated architectures such as Intserv [27], link buffer sizing has a different formulation because these architectures offer an application an ability to reserve buffer space at routers. Since the problem formulation intrinsically depends on the network architecture, an acceptable guideline for link buffer sizing in the Internet should be subject to the following constraint: a guideline for link buffer sizing has to be implementable without engaging Internet routers in any additional signaling.

The above constraint of using no additional signaling poses serious implementation challenges even for the existing guidelines. Some of the rules require from the router

to know the number of connections. However, IP does not provide routers with reliable means to acquire such knowledge. Albeit techniques exist for grouping datagrams into flows according to IP addresses, port numbers, and other header fields, it is difficult to use flow statistics to estimate accurately such quantities as the number of long file transfers over a specific TCP version [13]. It is even harder to implement guidelines involving the bitrate-delay product. IP offers no explicit mechanism for a router to learn round-trip propagation delays of passing traffic. Inference of these delays from datagram headers appears to be an extremely difficult, if not infeasible, task.

While the perfect knowledge of the needed parameters seems difficult to attain, approximate implementation of the guidelines is an option. Below, we discuss dangers hidden in this alternative. The bitrate-delay-product rule can be approximated by replacing the round-trip propagation delay with round-trip time (RTT), which includes both propagation and queueing. Consider a scenario where a bottleneck with bitrate B serves a single TCP NewReno connection that delivers a long file and operates in the congestion-avoidance mode. Round-trip propagation delay for the connection is D. The router infers average RTT of the connection precisely and sets the link buffer size to the product of B and RTT. The initial buffer size is set optimally to BD. Then, the connection load oscillates between BD and $2BD$, and queuing delay oscillates between 0 and D. Since average RTT becomes $1.5D$, the router increases the buffer size to $1.5BD$. The connection load now oscillates between $1.25BD$ and $2.5BD$, and queuing delay oscillates between $0.25D$ and $1.5D$. Since average RTT climbs to $1.875D$, the router raises the buffer size to $1.875BD$, and the vicious circle of unnecessary increases continues as the buffer size converges to $3BD$, which is three times larger than desired. Moreover, one can construct such multiple-bottleneck topologies that buffer sizes and queueing delays grow without bound if the buffers are sized with RTT instead of round-trip propagation delay.

Our discussion in the previous paragraph considers scenarios where routers automatically inflate their buffer sizes in proportion to increasing RTT. The same disastrous effects would occur but at a slower pace if the approximated guideline were carried out by human operators.

In the light of the exposed danger from replacing the propagation delay in a guideline with RTT, it is reasonable to question about current practices in buffer configuration in real networks. Anecdotal evidence suggests that router operators either allocate the whole buffer space provided by the manufacturer or set the buffer size to a bitrate-"delay" product where "delay" is an arbitrarily-chosen large constant, e.g., 500 ms or a transoceanic propagation time. As our analysis shows, choosing such huge buffers can lead to excessive queueing. End-to-end Internet measurements indirectly confirm the aforesaid practices and their consequences: the round-trip time of the same TCP connection can often vary by several seconds according to Aikat et al. [28].

6. Small Constant Buffer Sizes

Based on our prior argument that a buffer sizing guideline should reconcile needs of diverse Internet applications and be implementable within the Internet architecture, we now reformulate the problem of link buffer sizing as follows: *design a buffer sizing algorithm that accommodates needs of all Internet applications without engaging IP routers in any additional signaling.*

In Section 5, we discuss challenges imposed by the application diversity and implementability requirement, e.g., routers cannot compute round-trip propagation delays without extra signaling. Given the severe constraints, what might be an acceptable solution for the reformulated problem? Our proposal comes from the observation that small buffers are better suitable for reconciling needs of different applications. Whereas end systems have many options for dealing with link underutilization, an end system cannot remove queueing delay that other end systems create at a shared FIFO Droptail buffer. To make our proposal specific, we suggest the following guideline: *set the buffer size to $2L$ datagrams, where L is the number of input links.*

The factor of L in our guideline is for enabling the router to losslessly handle simultaneous datagram arrivals from each input link. We see no fundamental reason for favoring the coefficient of 2. However, overall expression $2L$ ensures that the buffer size is small enough to avoid significant queueing delays.

7. Congestion Control with Small Buffers

Validation of our new guideline is an important next step. It is also an extremely challenging step because as per our discussion in Section 4, the untrustworthiness of ns-2 and other existing simulation tools make it necessary to conduct the validation in real networks. As a first step toward the comprehensive evaluation, this section examines in more detail how negatively small buffers affect performance of long file transfers, what are sources of the negative impacts, and how alternative congestion control techniques are capable of providing high end-to-end goodput to long file transfers despite small link buffers.

We conduct a series of simulations in the same single-bottleneck network topology as in Figure 1a. The simulation setup is similar to the described in Section 3.1 except for the following details. The core link has bitrate 10 Mbps. The bitrate of each access link is x times larger; we refer to x as access link speedup. While the propagation delays of all access links are kept at 0.5 ms, propagation delay of the core link equals d. Hence, round-trip propagation delay is equal to $2d + 2$ ms. We examine two buffer sizes: (a) 250 datagrams, which is around thrice the bitrate-delay product for $d = 50$ ms, and (b) 10 datagrams as per our guideline of small buffers. In both settings, long file transfers rely on TCP NewReno and last 100 s.

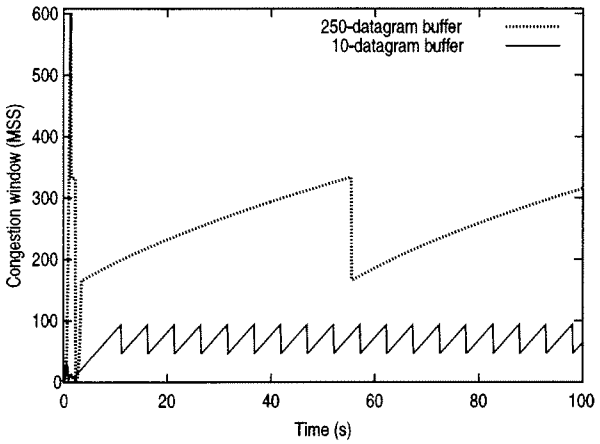

Figure 7. TCP NewReno dynamics with large and small buffers.

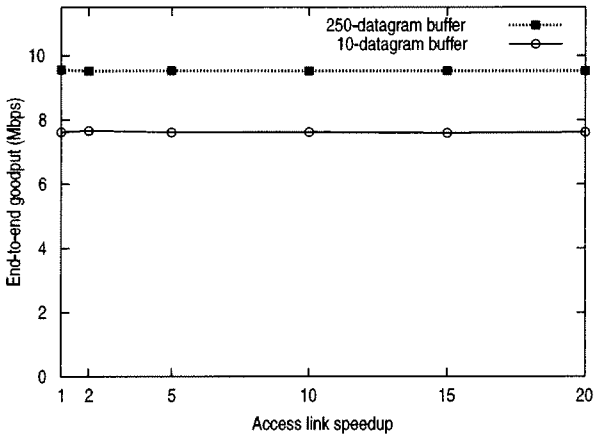

Figure 8. Independence of end-to-end goodput from access link speedup.

Figure 7 illustrates how the buffer size of the core bottleneck link affects the dynamics of a single file transfer when d = 50 ms and x = 10, i.e., propagation delay of the core link is 50 ms, and each access link has speedup 10 and thus bitrate 100 Mbps. In this topology, the bitrate-delay product equals 85 datagrams. With the 250-datagram link buffer, transmission exhibits a well-known pattern in which the number of unacknowledged maximum-size segments (MSS) overshoots the path capacity of 335 datagrams by a factor of almost 2 in slow start, causing massive datagram losses early on. Soon after, congestion avoidance ensues where the window oscillates between 167 and 334 MSS, persistently keeping at least 82 datagrams in the bottleneck link buffer and slowly increasing one-way delay to and beyond 350 ms, i.e., seven times its propagation component.

With the 10-datagram buffer, TCP NewReno dynamics are quite different. Since the access link speedup is high, each round of slow-start transmission injects a burst of data-grams into the network at an average rate of twice the bottleneck link bitrate, and the second half of the burst needs to be queued at the bottleneck link. Consequently, the transfer quickly exhausts the link buffer, loses a datagram, and abandons the slow start when the congestion window covers only 32 MSS, i.e., much less than needed to saturate the bottleneck link over entire round-trip propagation time. It is congestion avoidance that helps the sender to eventually reach the full link utilization. Then, the window oscillates between 47 and 94 MSS, i.e., between 55% utilization of the bottleneck link and 100% link utilization with up to 12 ms queueing delay. Note from Figure 7 that the queue never becomes large enough to reveal the window growth nonlinearity that manifests itself clearly during queue buildups in the 250-datagram buffer.

The inability of small link buffers to accommodate the bursts of datagrams during TCP slow start has a well-known solution of pacing that spreads data segments from the current window over the entire RTT [29–32]. However, the issue of sub-RTT burstiness (and hence its fixes) is of little significance when long file transfers operate mostly in congestion avoidance, as in the above experiment. Figure 8 supports this assertion by showing that end-to-end goodput of the single TCP flow remains almost constant as the access link speedup increases from 1 to 20.

Figure 8 suggests that the detrimental impact of small link buffers on goodput is primarily due to underutilization of the bottleneck link during congestion avoidance: end-to-end goodput of 7.6 Mbps with the small buffer versus 9.5 Mbps with the large buffer is consistent with the average utilizations that the congestion-avoidance mode maintains for the bottleneck link with the respective buffer sizes. The experimental results also agree with the classical AIMD control theory [7], which implies that even with a minimal buffer at the bottleneck link, AIMD(a;b) algorithm with increase step a = 1 MSS and decrease factor b = 0.5 oscillates between 100% and at least 50% of the path capacity, yielding at least 75% average utilization of the bottleneck link by TCP NewReno in congestion avoidance. To achieve a higher utilization, end systems can adopt another well-known technique of smoother congestion control. Examples of such smoother control include TCP Vegas, and other linear and nonlinear adjustment algorithms [33–39]. In particular, one can increase the average utilization of the bottleneck link in congestion avoidance to at least 94% by simply raising AIMD(a;b) decrease factor b from 0.5 to 0.875, as originally proposed by Ramakrishnan and Jain [40].

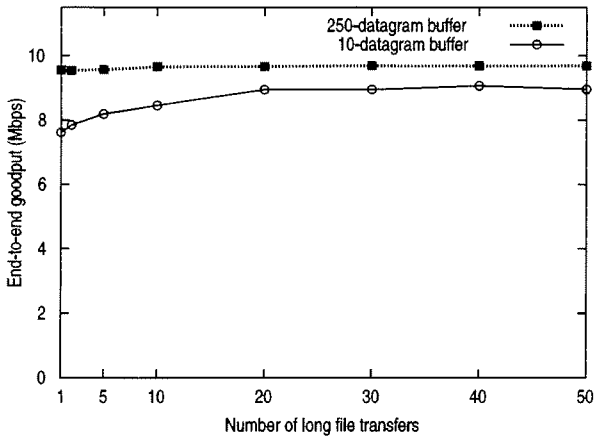

Figure 9. Benefits from increasing the number of connections: small windows with the 10-datagram link buffer improve overall goodput through loss desynchronization.

Another impact of small link buffers on behavior of TCP NewReno is in potential increase of datagram loss rates and timeout frequencies. To explore this issue, we conduct a series of simulations where d = 50 ms, x = 10, and the number of long file transfers varies from 1 to 50. With the 10-datagram link buffer, raising the number of connections to 50 reduces the per-connection share of the path capacity to less than 2 datagrams. Smaller achievable windows do icrease timeout frequencies and lead to severe starvation of some transfers when the number of connections is 20 or more. With 50 connections, about a half of them are completely shut down. Such unfairness is consistent with findings by Qiu et al. [21]. Figure 9 reveals another interesting result that the unfairness is accompanied by higher end-to-end goodput: while a single connection provides goodput of 7.6 Mbps, goodput grows to about 9 Mbps with 20 or more transfers. The underlying reason is loss desynchronization that reduces the amplitude of overall load oscillations after an initial transient and thereby enables TCP NewReno to boost its average utilization of the bottleneck link. End-to-end goodput also benefits (albeit less significantly) from shorter duration of the initial transient: a larger number of connections fill up the path capacity faster. This second contributor to the goodput improvement also affects the configuration with the 250-datagram buffer where goodput rises slightly as the number of connections grows from 1 to 10.

Figure 10 illustrates these two effects of unfairness and higher overall goodput more clearly. The reported experimental results are for twenty long file transfers, x = 10, and propagation delay d that varies from 1 to 50 ms. Figure 10a shows that as available windows shrink due to decreases in round-trip propagation delay, overall goodput with the 10-datagram buffer not only improves but also matches overall goodput supported by the 250-datagram buffer. The ability to match the performance of the 250-datagram buffer has a simple explanation: decreasing d reduces the path capacity, and 10 datagrams surpass the bitrate-delay product when d goes under 5 ms. Figure 10b demonstrates that while the 250-datagram buffer supports fair sharing of the bottleneck link capacity among all twenty flows, some transfers acquire unfairly high portions of the bottleneck link capacity at the expense of other transfers when the link buffer consists of only 10 datagrams. The unfairness is particular severe with d of 10 ms and lower; in these settings, about a half of all transfers are completely starved.

To resolve the unfairness caused by small link buffers, end-to-end congestion control can eliminate the source of the problem, i.e., the small size of windows. End systems can increase the window available to a connection by inserting delay into round-trip time, e.g., by delaying acknowledgments at the receiver [41, 42]. Such end-system queueing is preferable over traditional queueing at the shared buffer of a bottleneck link because queueing at an end system does not delay datagrams of other senders.

While the end-to-end congestion control techniques discussed in this section are potent building blocks for a protocol that serves long file transfers efficiently and fairly despite small link buffers, designing and tuning such a protocol is a challenging task. Recent work by Gu et al. [17] is a first solid step in addressing this challenge.

8. Conclusions

In this paper, we investigated the problem of how much buffer an IP router should allocate for its FIFO Droptail link. For a long time, setting the buffer size to the bitrate-delay product has been regarded as reasonable. Recent studies of interaction between queueing at IP routers and TCP congestion control proposed alternative guidelines. We used ns-2 simulations to explore and reconcile contradictions between the existing rules. Then, we argued that the problem of link buffer sizing needs a new formulation: *design a buffer sizing algorithm that accommodates needs of all Internet applications without engaging IP routers in any additional signaling.* Our proposal for solving the problem is to keep network queues short: *set the buffer size to 2L datagrams, where L is the number of input links.* We also discussed how end systems can utilize the network effectively in the presence of such small link buffers.

Simulation was not only a tool but also a subject of our studies. Our investigation showed the great importance of conducting simulations over the entire scope of realistic parameter settings, e.g., a wide range of values for the ratio of the bitrate-delay product to the number of TCP connections. We also discussed grievous concerns whether the universal network simulator ns-2 is a trustworthy evaluation platform for the problem of link buffer sizing. Dependable answers to the problem seem to require more

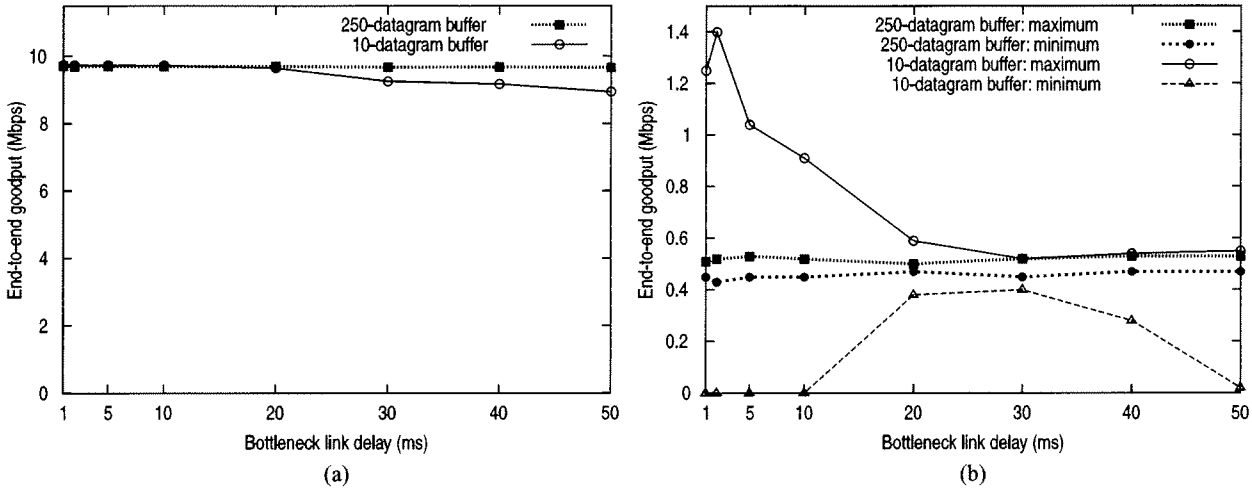

Figure 10. High overall goodput and worse fairness with small link buffers as round-trip propagation delay decreases: (a) Overall goodput; (b) Individual goodput.

detailed modeling of delays experienced by datagrams at routers and end systems. However, incorporation of more realistic models into simulations is in conflict with a clear need in improved scalability of the simulations. Our suggestion for overcoming this tension is to simulate link buffer sizing in a problem-specific model that represents important properties (e.g., of delay dynamics) precisely but makes simplifying assumptions about less relevant aspects (e.g., routing).

9. Acknowledgments

We would like to thank Guido Appenzeller, Konstantin Avrachenkov, Constantinos Dovrolis, Kevin Jeffay, Long Le, Dmitri Loguinov, Nick McKeown, and Don Towsley for comments and suggestions that helped us greatly in writing this paper.

10. References

[1] S. McCanne and S. Floyd, ns Network Simulator, www.isi.edu/nsnam/ns/

[2] University of Southern California. 1980. *DoD Standard Internet Protocol.* www.faqs.org/rfcs/rfc760.html, RFC 760.

[3] Postel, J. 1981. *Transmission Control Protocol.* www.faqs.org/rfcs/rfc793.html, RFC 793.

[4] Allman, M., V. Paxson, and W. Stevens. 1999. *TCP Congestion Control.* www.faqs.org/rfcs/rfc2581.html, RFC 2581.

[5] Jacobson, V. 1988. Congestion avoidance and control. In *Proceedings ACM SIGCOMM 1988*, ACM Press, New York, USA, pp. 314–329.

[6] Postel, J. 1980. *User Datagram Protocol.* www.faqs.org/rfcs/rfc768.html, RFC 768.

[7] Chiu, D. and R. Jain. 1989. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN*, 17(1):1–14.

[8] Jacobson, V. 1990. *Modified TCP Congestion Control Algorithm.* End2end-interest mailing list.

[9] Appenzeller, G., I. Keslassy, and N. McKeown. 2004. Sizing router buffers. In *Proceedings ACM SIGCOMM 2004*, ACM Press, New York, pp. 281–292.

[10] Wischik, D. and N. McKeown. 2005. Part I: buffer sizes for core routers. *ACM Computer Communication Review*, 35(3):75–78.

[11] Dhamdhere, A., H. Jiang, and C. Dovrolis. 2005. Buffer sizing for congested Internet links. In *Proceedings IEEE INFOCOM 2005*, IEEE Communications Society, New York, pp. 1072–1083.

[12] Le, L., K. Jeffay, and D. Smith. 2005. *Sizing Router Buffers for Application Performance.* Technical report. UNCCS-TR05-111. Department of Computer Science, University of North Carolina.

[13] Morris, R. 2000. Scalable TCP congestion control. In *Proceedings IEEE INFOCOM 2000*, IEEE Communications Society, New York, pp. 1176–1183.

[14] Gorinsky, S., A. Kantawala, and J. Turner. 2005. Link buffer sizing: a new look at the old problem. In *Proceedings IEEE Symposium on Computers and Communications (ISCC 2005)*, IEEE Computer Society, Washington, pp. 507–514.

[15] Enachescu, M., Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden. 2005. Part III: routers with very small buffers. *ACM Computer Communication Review*, 35(3):83–90.

[16] Raina, G., D. Towsley, and D. Wischik. 2005. Part II: control theory for buffer sizing. *ACM Computer Communication Review*, 35(3):79–82.

[17] Gu, Y., D. Towsley, C. Hollot, and H. Zhang. 2007. Congestion control for small buffer high speed networks. In *Proceedings IEEE INFOCOM 2007*, IEEE Communications Society, New York, pp. 1037–1045.

[18] Villamizar, C. and C. Song. 1994. High performance TCP in the ANSNET. *ACM SIGCOMM Computer Communication Review*, 24(5):45–60.

[19] Floyd, S. and T. Henderson. 1999. *The NewReno Modification to TCP's Fast Recovery Algorithm.* www.faqs.org/rfcs/rfc2582.html, RFC 2582.

[20] Sun, J., M. Zukerman, K.-T. Ko, G. Chen, and S. Chan. 2004. Effect of large buffers on TCP queueing behavior. In *Proceedings IEEE INFOCOM 2004*, IEEE Communications Society, New York, pp. 751–761.

[21] Qiu, L., Y. Zhang, and S. Keshav. 2001. Understanding the performance of many TCP flows. *Computer Networks*, 37(3-4):277–306

[22] Floyd, S. and E. Kohler. 2002. Internet research needs better models. In *Proceedings HotNets-I*, ACM Press, New York, pp. 29–34.

[23] Rizzo, L. 1997. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41.

[24] Brakmo, L., S. O'Malley, and L. Peterson. 1994. TCP Vegas: new techniques for congestion detection and avoidance. In *Proceedings ACM SIGCOMM 1994*, ACM Press, New York, pp. 24–35.

[25] Altman, E., K. Avrachenkov, and C. Barakat. 2002. TCP network calculus: the case of large delay-bandwidth product. In *Proceedings IEEE INFOCOM 2002*, IEEE Communications Society, New York, pp. 417–426.

[26] Avrachenkov, K., U. Ayesta, E. Altman, P. Nain, and C. Barakat. 2002. The effect of router buffer size on the TCP performance. In *Proceedings LONIIS Workshop on Telecommunication Networks and Teletraffic Theory*, LONIIS, St. Petersburg, Russia, pp. 116–121.

[27] Braden, R., D. Clark, and S. Shenker. 1994. *Integrated Services in the Internet Architecture: an Overview*. www.faqs.org/rfcs/rfc1633.html, RFC 1633.

[28] Aikat, J., J. Kaur, D. Smith, and K. Jeffay. 2003. Variability in TCP round-trip times. In *Proceedings ACM SIGCOMM Internet Measurement Conference*, ACM Press, New York, pp. 279–284.

[29] Aggarwal, A., S. Savage, and T. Anderson. 2000. Understanding the performance of TCP pacing. In *Proceedings IEEE INFOCOM 2000*, IEEE Communications Society, New York, pp. 1157–1165.

[30] Kulik, J., R. Coulter, D. Rockwell, and C. Partridge. 1999. Paced TCP for high delay-bandwidth networks. In *Proceedings IEEE Globecom 1999*, IEEE Communications Society, New York, pp. 56–65.

[31] Kulik, J., R. Coulter, D. Rockwell, and C. Partridge. 2000. *A Simulation Study of Paced TCP*. Technical report. CR-2000-209416, BBN Technologies, Cambridge, MA.

[32] Zhang, L., S. Shenker, and D. Clark. 1991. Observations and dynamics of a congestion control algorithm: the effects of two-way traffic. In *Proceedings ACM SIGCOMM 1991*, ACM Press, New York, pp. 133–147.

[33] Attie, P., A. Lahanas, and V. Tsaoussidis. 2003. Beyond AIMD: explicit fair-share calculation. In *Proceedings IEEE Symposium on Computers and Communications (ISCC 2003)*, IEEE Computer Society, Washington, pp. 727–734.

[34] Bansal, D. and H. Balakrishnan. 2001. Binomial congestion control algorithms. In *Proceedings IEEE INFOCOM 2001*, IEEE Communications Society, New York, pp. 631–640.

[35] Floyd, S., M. Handley, and J. Padhye. 2000a. *A Comparison of Equation-Based and AIMD Congestion Control*. www.aciri.org/tfrc/.

[36] Floyd, S., M. Handley, J. Padhye, and J. Widmer. 2000b. Equation-based congestion control for unicast applications. In *Proceedings ACM SIGCOMM 2000*, ACM Press, New York, pp. 43–56.

[37] Loguinov, D. and H. Radha. 2002. Increase-decrease congestion control for real-time streaming: scalability. In *Proceedings IEEE INFOCOM 2002*, IEEE Communications Society, New York, pp. 525–534.

[38] Podlesny, M. and S. Gorinsky. 2007a. MCP: Few bits for fairing and small queues in the stable state. In *Proceedings IEEE Symposium on Computers and Communications (ISCC 2007)*, IEEE Computer Society, Washington, pp. 1079–1084.

[39] Podlesny, M. and S. Gorinsky. 2007b. Multimodal congestion control for low stable-state queuing. In *Proceedings IEEE INFOCOM 2007*, IEEE Communications Society, New York, pp. 2466–2470.

[40] Ramakrishnan, K. and R. Jain. 1988. A binary feedback scheme for congestion avoidance in computer networks with connectionless network layer. In *Proceedings ACM SIGCOMM 1988*, ACM Press, New York, pp. 138–156.

[41] Blandford, K., S. Goldman, S. Gorinsky, Y. Zhou, and D. Dooly. 2007. Smartacking: improving TCP performance from the receiving end. *Journal of Internet Engineering*, 1(1):6–21.

[42] Clark, D. 1982. *Window and Acknowledgement Strategy in TCP*. www.faqs.org/rfcs/rfc813.html, RFC 813.

Sergey Gorinsky *is a native of Skhodnya, Russia. He received a degree of Engineer at Moscow Institute of Electronic Technology, Zelenograd, Russia and M.S. and Ph.D. degrees from the University of Texas at Austin, USA. Dr. Gorinsky is currently with Washington University in St. Louis, USA where he works as an Assistant Professor at the Applied Research Laboratory in the Department of Computer Science and Engineering. His primary research interests are in computer networking and distributed systems. Dr. Gorinsky's work has appeared at top conferences and journals such as ACM SIGCOMM, IEEE INFOCOM, and IEEE/ACM Transactions on Networking. He has been serving on Technical Program Committees of IEEE INFOCOM and other networking conferences.*

Anshul Kantawala *received a B.S. degree in Computer Science from Washington University in St. Louis, M.S. degree in Computer and Information Science from the University of Pennsylvania in Philadelphia and a D.Sc. degree from Washington University in St. Louis. Dr. Kantawala is currently working in Lockheed Martin, and his primary research interests are in mobile communications, satellite networking, and transport protocols. The work presented in this paper was performed while Dr. Kantawala was at the Applied Research Laboratory in Washington University in St. Louis.*

Jonathan S. Turner *received M.S. and Ph.D. degrees in Computer Science from Northwestern University in 1979 and 1981. He holds the Barbara and Jerome Cox Chair of Computer Science at Washington University, serves as the Chairman of the Department of Computer Science and Engineering, and is the Director of the Applied Research Laboratory (ARL). ARL creates experimental networking technology to validate and demonstrate new research innovations. Current projects at ARL center on the design of high-performance virtualized network platforms and clean-slate network architectures. Professor Turner served as the Chief Scientist for Growth Networks, a startup company that developed scalable switching components for Internet routers and ATM switches before being acquired by Cisco Systems in early 2000. Turner is a fellow of both the ACM and the IEEE and member of the National Academy of Engineering. He received the Koji Kobayashi Computers and Communications Award from the IEEE in 1994 and the IEEE Millenium Medal in 2000. He has been awarded 30 patents for his work on switching systems and has many widely cited publications.*