Link Buffer Sizing: A New Look at the Old Problem

Sergey Gorinsky Anshul Kantawala Jonathan Turner

Applied Research Laboratory, Department of Computer Science and Engineering
Washington University, St. Louis, MO 63130-4899, USA

{*gorinsky, anshul, jst*}*@arl.wustl.edu*

Abstract

We revisit the question of how much buffer an IP router should allocate for its Droptail FIFO 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 offered alternative guidelines. First, we explore and reconcile contradictions between the existing rules. Then, we argue 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 solution keeps network queues short: set the buffer size to $2L$ datagrams, where L is the number of input links. We also explain how end systems can utilize the network effectively despite such small buffering at routers.

1 Introduction

Simplicity of IP (Internet Protocol) [23] is often cited as the primary reason for the explosive growth of the Internet. IP merely defines a format of datagrams. 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. Apart from enabling the forwarding function, IP networks require no signaling between routers.

Although the minimal signaling promotes quick deployment, IP networks can suffer from congestion when a router has to buffer or discard datagrams destined for a busy output link. To avoid persistent congestion, end systems are expected to curb their transmission upon inferring congestion from implicit signs such as datagram loss. Many applications rely on TCP (Transmission Control Protocol) [19] for congestion control. Other applications communicate over UDP (User Datagram Protocol) [18] and need to control congestion on their own.

End-to-end protocols for congestion control are based on perpetual probing for available capacity. Even with a static set of end-to-end connections, their total load on a shared bottleneck link does not converge to a single value but oscillates within a range [8]. If the bottleneck link has a small buffer, the oscillations can lead to incomplete utilization of the link. If the buffer is large, queueing at the link can create unnecessary delays.

The question of how much buffer an IP router should allocate for its Droptail FIFO link is traditionally formulated as a problem of fully utilizing the bottleneck link without excessive queueing. For a long time, setting the buffer size to the product of the link bitrate and round-trip propagation delay was considered reasonable. More elaborate recent analyses indicate that the traditional rule of thumb should be adjusted. One guideline suggests decreasing the buffer size as the number of connections increases. Other proposals argue that the optimal buffer size grows proportionally to the number of connections.

In this paper, we explore contradictions between existing guidelines for link buffer sizing. After tracing the contradictions to differences in goals and assumptions, we derive a reconciling rule. Then, we argue that the new guideline is also inadequate because it solves an incorrectly formulated problem. Our new problem formulation for link buffer sizing requires from an acceptable solution to accommodate needs of all Internet applications and be implementable within the Internet architecture. We propose such a solution, which prescribes small buffers. We conclude by explaining how end systems can utilize the network effectively without large buffering at routers.

The rest of the paper is structured as follows. Section 2 reviews existing guidelines. Their contradictions are explained and reconciled in Section 3. Section 4 argues that Internet application diversity and guideline implementability require a new problem formulation for link buffer sizing. Section 5 presents our solution for the reformulated problem. Section 6 finishes the paper with a summary.

2 Existing guidelines for link buffer sizing

The traditional rule of thumb prescribes setting the buffer size to the **bitrate-delay product**. Whereas the guideline is often attributed to Villamizar and Song [24], the rule has been priorly suggested by others, e.g., Jacobson [13]. The bitrate-delay-product guideline is easily derivable from a model where the bottleneck link has a Droptail FIFO buffer and serves only one 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 [8]. The assumed additive-increase multiplicative-decrease (AIMD) pattern approximates well the congestion-avoidance mode in such versions of TCP as Reno [13] and NewReno [12]. 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 bitrate and round-trip propagation delay of the connection equals B, then the decrease drains the buffer completely without underutilizing the link. Since the buffer size in this setting also equals B, the analysis leads to the guideline of setting the link buffer size to the bitrate-delay product.

The above simple model considers only one TCP connection. Recent analyses of models where a bottleneck link serves multiple connections have yielded two mutually contradictory guidelines: over-square-root and connection-proportional allocation.

Over-square-root generalizes the traditional guideline by prescribing to set the buffer size to the bitrate-delay product divided by $\sqrt{n}$ where n is the number of TCP connections [3]. The refinement is derived from evidence that some connections might 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 [3, 22]. Hence, the over-square-root guideline proposes decreasing the buffer size as the number of TCP connections increases.

Connection-proportional allocation takes an opposite view and suggests that the buffer size should be proportional to the number of TCP connections [10, 14, 16]. This guideline stems from an observation that the connections can suffer from high datagram losses and frequent retransmission timeouts if the link buffer does not accommodate at least few datagrams per connection.

3 Contradictions reconciled

Although over-square-root and connection-proportional allocation agree that the traditional rule is inadequate for scenarios with multiple connections, the new guidelines conflict on whether the buffer size should be decreased or increased. Furthermore, connection-proportional allocation does not scale up the buffer as the link bitrate or network propagation delays increase. In this section, we explore and reconcile the contradictions between the guidelines.

3.1 Experimental validation

Since practice is the best judge of theory, we check validity of the guidelines by conducting ns-2 [17] simulations in a traditional single-bottleneck topology shown in Figure 1a. End systems S_i and D_i host respectively sending and receiving ends of unicast applications; with i varying from 1 to N, the network contains $2N$ end systems. All the applications communicate data via datagrams of size 1500 bytes. 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 larger than the bottleneck bitrate. Every link uses FIFO Droptail buffering.

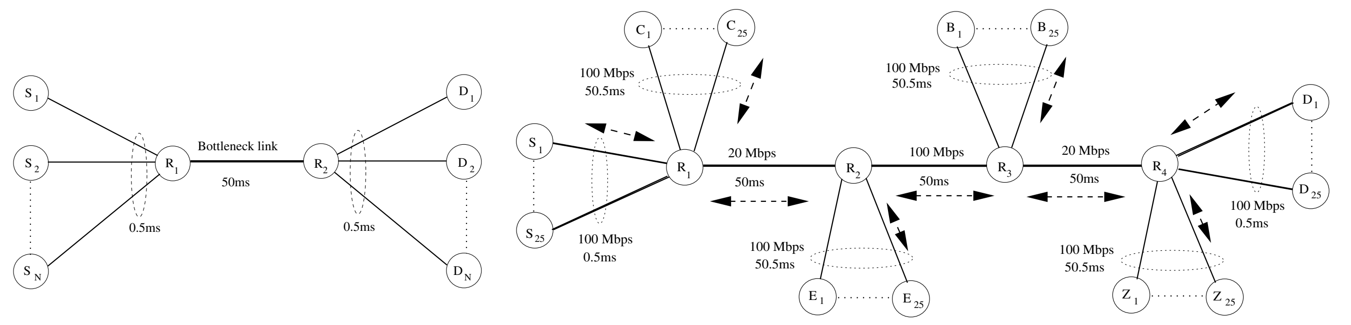
In the first series of our experiments, each of the N applications transfers a long file over TCP NewReno. We repeat the experiments for the following three values of the bottleneck bitrate: 100 Mbps, 10 Mbps, and 1 Mbps.

For file transfers, full utilization of the bottleneck link is not a goal in itself. What is important for this application is *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. We perform all the measurements over the whole experiment duration of 200 seconds.

Figures 2 and 3 present our results. 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. Instead of remaining constant at the bitrate-delay product, the optimal buffer size starts growing. The growth is approximately linear and therefore consistent with 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 experiments validates over-square-root that prescribes decreasing the buffer size as the number of connections grows.

3.2 Analysis and reconciliation

First of all, why do our experiments show no reasons to follow over-square-root? The guideline is derived from a model where tens of thousands connections share a back-



(a) Single-bottleneck topology (b) Multiple-bottleneck topology

Figure 1. Network topologies in our experiments

bone bottleneck link. Besides, Appenzeller, Keslassy, and McKeown [3] indicate that desynchronization of load reductions appears and reveals the $\sqrt{n}$ adjustment only when n is at least on the order of thousand connections. Hence, checking validity of over-square-root requires experiments in a high-speed network with thousands of concurrent connections. Unfortunately, we were not able to complete such simulations in ns-2. Furthermore, the scenario assumed by the over-square-root model is not easy to find in the modern Internet where bottlenecks are typically not backbone links but slower access links serving fewer connections.

Whereas our results offer no sound basis for judging correctness of over-square-root for a backbone bottleneck link, 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; however, over-square-root sets the buffer size to 10% of the bitrate-delay product and leads to underutilization of the bottleneck link and thereby to lower goodput of the file transfers.

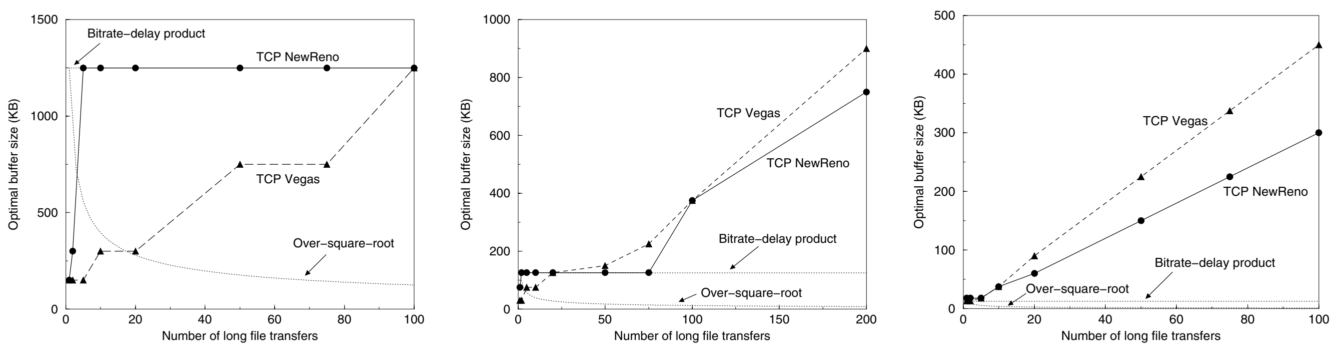
With the $\sqrt{n}$ refinement out of the way, let us now focus on the contradiction between the bitrate-delay product and connection-proportional allocation. Figure 2 indicates that validity of these guidelines depends on the ratio of the bitrate-delay product to the number of TCP NewReno connections. 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 connection-proportional allocation. What makes the ratio such an important parameter? Our analysis confirms correctness of logical derivations behind both guidelines. The contradictory conclusions are due to differences in goals and assumptions. The original rule strives to utilize the bottleneck link fully and assumes that a TCP connection operates uninterruptedly in the congestion-avoidance mode. If the bottleneck link buffer does not accommodate at least few datagrams for the connection, the assumption can be wrong because high loss rates cause frequent retransmission timeouts. On the other hand, connection-proportional allocation pursues an objective of keeping each connection in the congestion-avoidance mode but does not concern itself with the bottleneck link utilization. Hence, if the ratio of the bitrate-delay product to the number of connections is high, connection-proportional allocation underutilizes the link and thereby fails to maximize end-to-end goodput.

Since the contradictions between the existing guidelines are due to differences in their assumptions and goals, we can derive a consistent guideline from a model that combines the objectives of keeping each TCP connection in the congestion-avoidance mode and utilizing the bottleneck link fully: *set the buffer size to the maximum of the bitrate-delay product and connection-proportional allocation*.

4 New problem formulation

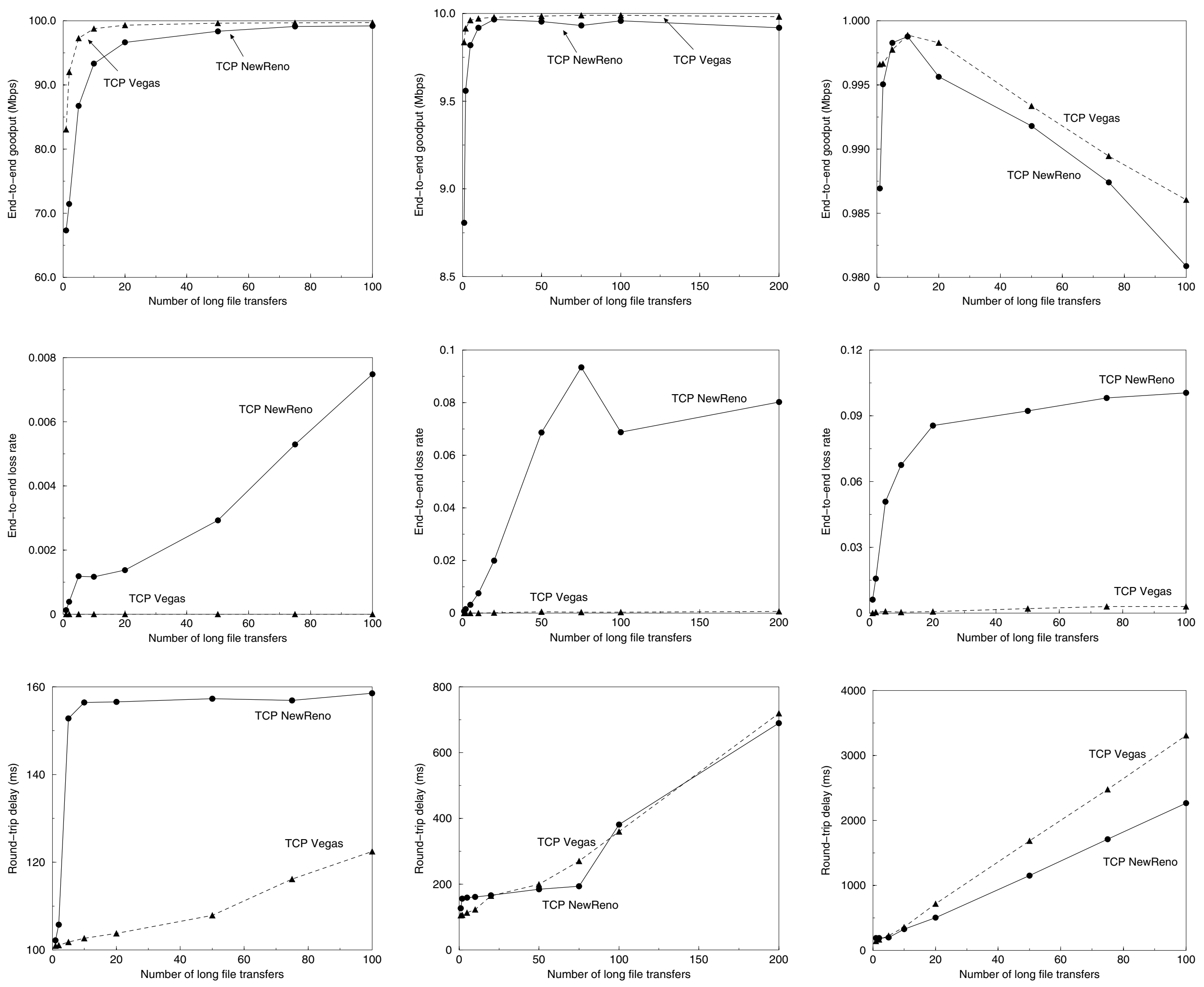
In Section 3, we derived a reconciling guideline. Is our guideline *the* final word in link buffer sizing? Even we answer this question not affirmatively. First, if desynchronization of load reductions does occur with thousands of TCP connections, the link-utilization component of the guideline (i.e., bitrate-delay product) has to be amended respectively. However, a more fundamental reason exists for discounting our new guideline as well as the guidelines it reconciles – *we are solving an incorrectly formulated problem.*

Figure 2 hints at this flaw in the current approach by demonstrating that the optimal buffer size for Vegas [7] 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 AIAD (additive-increase additive-decrease) 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 a traditional formulation of the buffer sizing problem: *find the buffer size that optimizes performance of long file transfers over a version of TCP*.



(a) 100 Mbps bottleneck link (b) 10 Mbps bottleneck link (c) 1 Mbps bottleneck link

Figure 2. Optimal buffer size for the file transfers over TCP



(a) 100 Mbps bottleneck link (b) 10 Mbps bottleneck link (c) 1 Mbps bottleneck link

Figure 3. End-to-end goodput, loss rate, and round-trip delay when the link buffer size is optimal

We argue that the traditional problem formulation is too limited. Buffer sizing should accommodate not only a particular 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. Sections 4.1 and 4.2 discuss these two issues below.

4.1 Diversity of Internet applications

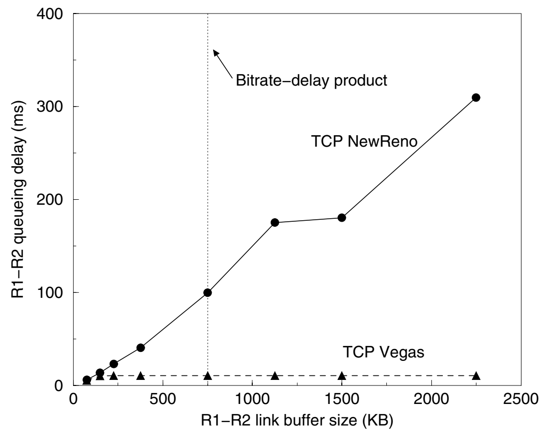
Long file transfers over TCP are common in the Internet. However, the Internet also serves other traffic such as short file transfers over TCP and interactive streaming over UDP. Whereas link buffer sizing affects these other applications as well, the impact is qualitatively different.

Unlike with large files, short file transfers over TCP do not operate in the congestion-avoidance mode for long. 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. Recent studies [2, 3, 4] show that a small constant buffer is sufficient for optimal performance of short file transfers regardless of their number.

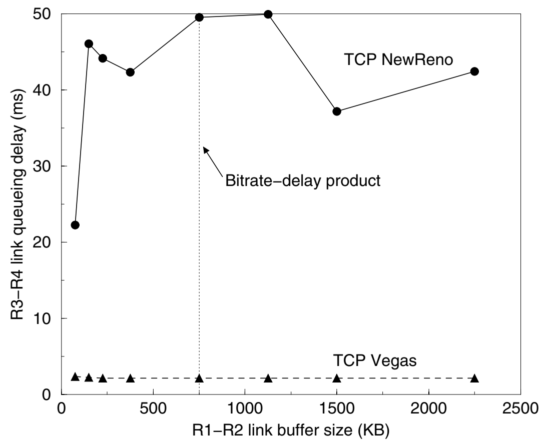
For interactive streaming over UDP, queueing delays are even more consequential since interaction at round-trip times larger than few hundred milliseconds makes humans uncomfortable. Whereas such applications reject TCP due to its jittery transmission and extra delay of its reliable in-order delivery, communication over UDP comes with responsibility of performing own congestion control. What a particular streaming application views as an optimal buffer size depends greatly on its algorithm for congestion control.

Sizing a link buffer to optimize performance of long file transfers can cause huge queueing delays, such as several seconds shown in Figure 3c. As we discuss above, extensive queueing can 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-bit-rate 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 second. All the applications use datagrams 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, we measure link loss rates, queueing delays, and utilizations over the whole experiment duration of 200 seconds.

Figure 4 reports queueing delays for links R_1-R_2 and R_3-R_4 when 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 meaningful for each type of the examined appli-

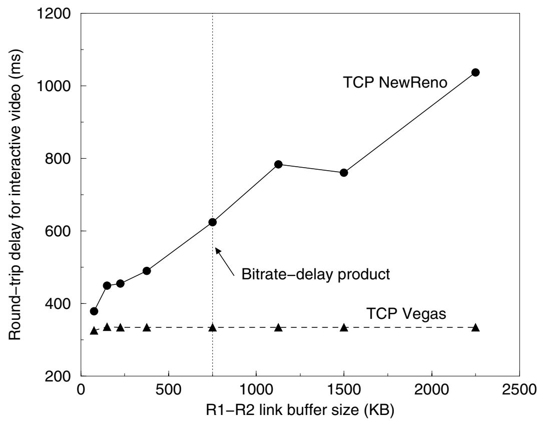


(a) Link R_1-R_2

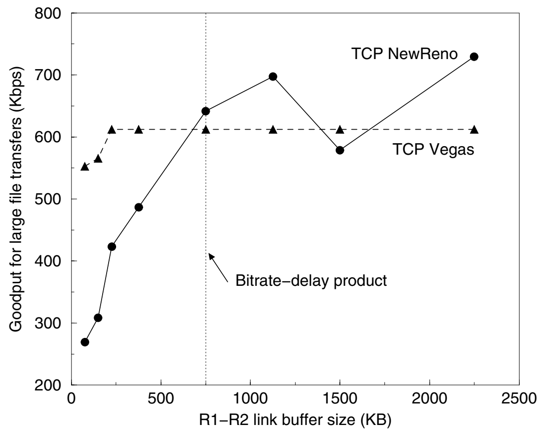


(b) Link R_3-R_4

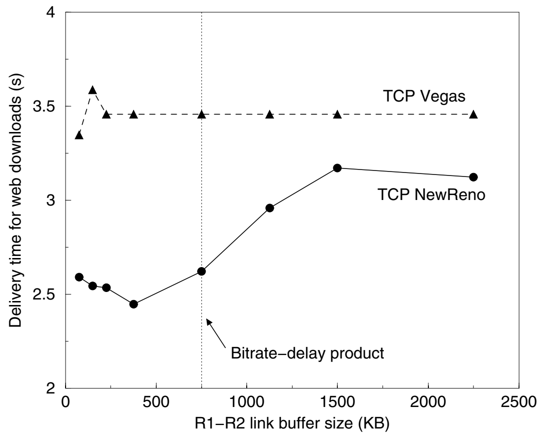
Figure 4. Link queueing delays in the multiple-bottleneck topology



(a) Round-trip time of interactive video



(b) End-to-end goodput of long file transfers



(c) Delivery time of short web downloads

Figure 5. Performance of different applications in the multiple-bottleneck topology

cations: 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. However, the bitrate-delay-product setting inflates the round-trip time above 600 ms, which is unacceptably large for interactive video.

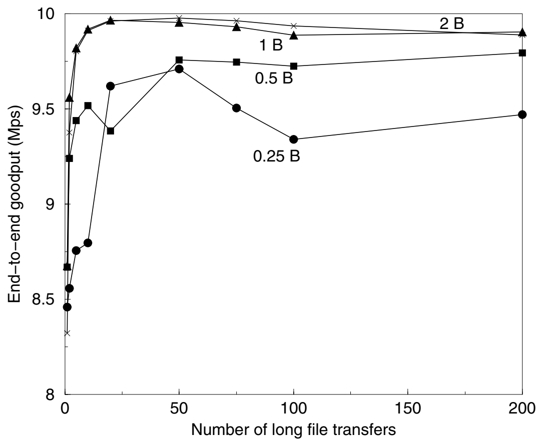


Figure 6. Impact of suboptimal buffer sizes on long file transfers over TCP NewReno

The above experiments show that choosing a large buffer to accommodate long file transfers can lethally hurt interactive streaming. Can a smaller buffer reconcile needs of all the applications? Earlier studies reply affirmatively by demonstrating that small buffer sizes do not disrupt file transfers substantially in terms of average goodput, even though individual goodputs can become more variable [4, 20]. To examine 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 B. As expected, Figure 6 shows that end-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, selecting a significantly lower buffer size can provide long file transfers with reasonably good performance.

4.2 Implementability of sizing guidelines

Section 4.1 indicates that reconciling diverse needs of Internet applications favors small buffers. Before we describe our specific solution for link buffer sizing in Section 5, we now discuss importance and difficulties of implementing a sizing guideline within the Internet architecture.

Simplicity of the Internet architecture plays an implicit but major role in traditional formulations of the buffer sizing problem. Due to minimality 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 [6], 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 has to *be implementable without engaging IP routers in any additional signaling*.

The constraint of 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 [16]. It is even harder to implement guidelines involving the *bitrate-delay product*: whereas IP offers no explicit mechanism for a router to learn round-trip propagation delays of passing traffic, inferring these delays from datagram headers seems all but infeasible.

Let us now study an option of implementing the guidelines approximately. 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 link 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 queueing delay oscillates between 0 and D. Since average RTT becomes $1.5D$, the router increases the buffer size to $1.5BD$. The connection load oscillates now between $1.25BD$ and $2.5BD$, and queueing 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. Furthermore, it is easy to construct multiple-bottleneck topologies where sizing the buffers with RTT instead of round-trip propagation delay leads to unbounded growth of buffer sizes and queueing delays.

In the above examples, routers automatically inflate their buffer sizes in proportion to increasing RTT. Note that if the approximated guideline is carried out by humans, the same disastrous effects ensue, albeit at a slower pace.

Whereas we clearly demonstrate that replacing the propagation delay in a guideline with RTT can be extremely dangerous, it is reasonable to inquire about current practices in buffer configuration. 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: Aikat, Kaur, Smith, and Jeffay [1] report that RTT of a TCP connection can vary by several seconds.

5 Solving the reformulated problem

Section 4 argues that a buffer sizing guideline should reconcile needs of diverse Internet applications and be implementable within the Internet architecture. Hence, we 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.*

Section 4 also shows challenges posed by the application diversity and implementability requirement, e.g., without extra signaling, routers cannot compute round-trip propagation delays. 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 Droptail FIFO 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 reference to 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.

Anticipating a valid criticism that the small buffer size increases TCP loss rates and timeout frequencies, we now elaborate on how end systems can utilize the network effectively without large buffering at routers. Arguments for connection-proportional allocation are based on the fact that to operate in the desirable congestion-avoidance mode, a TCP connection should have RTT that corresponds to a window of at least few data segments. However, the performance-boosting delay does not have to added by the network. Instead, end systems can do the needed queueing, e.g., by delaying acknowledgments [5, 9]. The end-system option is actually more preferable because it does not impose the delay on the other traffic.

End systems can address also concerns about link underutilization. With the minimal buffer at the bottleneck link, TCP NewReno in the congestion-avoidance mode provides at least 75% utilization of the link because the load oscillates between 100% and at least 50% of the link bitrate. To achieve higher utilization, end systems can adopt smoother congestion control such as TCP Vegas or smooth UDP-based alternatives [11, 15]. For example, merely changing the AIMD decrease factor from 0.5 to 0.875 [21] boosts the bottleneck link utilization to at least 94%.

6 Conclusion

This paper revisited the old question of how much buffer an IP router should allocate for its Droptail FIFO 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. In this paper, we explored and reconciled 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 solution keeps network queues short: *set the buffer size to* $2L$ *datagrams, where* L *is the number of input links*. We also explained how end systems can utilize the network effectively despite such small buffering at routers.

Acknowledgments

We would like to thank Dmitri Loguinov, Konstantin Avrachenkov, and anonymous ISCC 2005 reviewers for comments and suggestions that greatly helped us to prepare the improved final version of this paper.

References

[1] J. Aikat, J. Kaur, D. Smith, and K. Jeffay. Variability in TCP Round-trip Times. In *Proceedings ACM SIGCOMM Internet Measurement Conference*, October 2003.

[2] E. Altman, K. Avrachenkov, and C. Barakat. TCP Network Calculus: The Case of Large Delay-Bandwidth Product. In *Proceedings IEEE INFOCOM 2002*, July 2002.

[3] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing Router Buffers. In *Proceedings ACM SIGCOMM 2004*, September 2004.

[4] K. Avrachenkov, U. Ayesta, E. Altman, P. Nain, and C. Barakat. The Effect of Router Buffer Size on the TCP performance. In *Proceedings LONIIS Workshop on Telecommunication Networks and Teletraffic Theory*, January 2002.

[5] K. Blandford, S. Goldman, S. Gorinsky, Y. Zhou, and D. Dooly. Smartacking: Improving TCP Performance from the Receiving End. Technical Report WUCSE-2005-4, *www.arl.wustl.edu/~gorinsky/pdf/WUCSE-TR-2005-4.pdf*, Department of Computer Science and Engineering, Washington University in St. Louis, January 2005.

[6] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. RFC 1633, June 1994.

[7] L. Brakmo, S. O'Malley, and L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proceedings ACM SIGCOMM 1994*, August 1994.

[8] D. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[9] D. Clark. Window and Acknowledgement Strategy in TCP. RFC 813, July 1982.

[10] A. Dhamdhere, H. Jiang, and C. Dovrolis. Buffer Sizing for Congested Internet Links. In *Proceedings IEEE INFOCOM 2005*, March 2005.

[11] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. In *Proceedings ACM SIGCOMM 2000*, August 2000.

[12] S. Floyd and T. Henderson. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 2582, April 1999.

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

[14] L. Le, K. Jeffay, and D. Smith. Sizing Router Buffers for Application Performance. Technical Report UNC-CS-TR05-111, Department of Computer Science, University of North Carolina, *www.cs.unc.edu/~le/papers/sizing-buffers.pdf*, January 2005.

[15] D. Loguinov and H. Radha. Increase-Decrease Congestion Control for Real-time Streaming: Scalability. In *Proceedings IEEE INFOCOM 2002*, June 2002.

[16] R. Morris. Scalable TCP Congestion Control. In *Proceedings IEEE INFOCOM 2000*, March 2000.

[17] UCB/LBNL/VINT Network Simulator ns-2. www-mash.cs.berkeley.edu/ns, May 2004.

[18] J. Postel. User Datagram Protocol. RFC 768, October 1980.

[19] J. Postel. Transmission Control Protocol. RFC 793, September 1981.

[20] L. Qiu, Y. Zhang, and S. Keshav. Understanding the Performance of Many TCP Flows. *Computer Networks*, 37(3-4):277–306, November 2001.

[21] K. Ramakrishnan and R. Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with Connectionless Network Layer. In *Proceedings ACM SIGCOMM 1988*, August 1988.

[22] J. Sun, M. Zukerman, K.-T. Ko, G. Chen, and S. Chan. Effect of Large Buffers on TCP Queueing Behavior. In *Proceedings IEEE INFOCOM 2004*, March 2004.

[23] Univeristy of Southern California. DoD Standard Internet Protocol. RFC 760, January 1980.

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