Report Number: WUCSE-2004-82

2004-12-04

Link Buffer Sizing: a New Look at the Old Problem

Sergey Gorinsky, Anshul Kantawala, and Jonathan Turner

In this paper, we revisit the question of how much buffer an IP router should allocate for its output link. For a long time, the intuitive answer of setting the buffer size to the bitrate-delay product has been widely regarded as reasonable. Recent studies of interaction between queueing at IP routers and TCP congestion control proposed alternative answers. First, we expose and explain contradictions between existing guidelines for link buffer sizing. Then, we argue that the problem of link buffer sizing needs a different formulation. In particular, the chosen buffer size should accommodate not only common versions of TCP... **Read complete abstract on page 2.**

Link Buffer Sizing: a New Look at the Old Problem

Sergey Gorinsky, Anshul Kantawala, and Jonathan Turner

Complete Abstract:

In this paper, we revisit the question of how much buffer an IP router should allocate for its output link. For a long time, the intuitive answer of setting the buffer size to the bitrate-delay product has been widely regarded as reasonable. Recent studies of interaction between queueing at IP routers and TCP congestion control proposed alternative answers. First, we expose and explain contradictions between existing guidelines for link buffer sizing. Then, we argue that the problem of link buffer sizing needs a different formulation. In particular, the chosen buffer size should accommodate not only common versions of TCP but also UDP traffic. Besides, our new formulation of the problem contains an explicit constraint of not engaging IP routers in any additional signaling. We conclude the paper by outlining a promising direction for solving the reformulated problem.

Department of Computer Science & Engineering

Washington University in St. Louis
School of Engineering & Applied Science

2004-82

Link Buffer Sizing: a New Look at the Old Problem

Authors: Gorinsky, Sergey; Kantawala, Anshul; Turner, Jonathan

December 4, 2004

Abstract: In this paper, we revisit the question of how much buffer an IP router should allocate for its output link. For a long time, the intuitive answer of setting the buffer size to the bitrate-delay product has been widely regarded as reasonable. Recent studies of interaction between queueing at IP routers and TCP congestion control proposed alternative answers. First, we expose and explain contradictions between existing guidelines for link buffer sizing. Then, we argue that the problem of link buffer sizing needs a different formulation. In particular, the chosen buffer size should accommodate not only common versions of TCP but also UDP traffic. Besides, our new formulation of the problem contains an explicit constraint of not engaging IP routers in any additional signaling. We conclude the paper by outlining a promising direction for solving the reformulated problem.

Link Buffer Sizing: a New Look at the Old Problem

Sergey Gorinsky, Anshul Kantawala, and Jonathan Turner
{*gorinsky, anshul, jst*}*@arl.wustl.edu*

WUCSE-2004-82

December 4, 2004

Department of Computer Science and Engineering
Campus Box 1045
Washington University
One Brookings Drive
St. Louis, MO 63130-4899

Abstract

In this paper, we revisit the question of how much buffer an IP router should allocate for its output link. For a long time, the intuitive answer of setting the buffer size to the bitrate-delay product has been widely regarded as reasonable. Recent studies of interaction between queueing at IP routers and TCP congestion control proposed alternative answers. First, we expose and explain contradictions between existing guidelines for link buffer sizing. Then, we argue that the problem of link buffer sizing needs a different formulation. In particular, the chosen buffer size should accommodate not only common versions of TCP but also UDP traffic. Besides, our new formulation of the problem contains an explicit constraint of not engaging IP routers in any additional signaling. We conclude the paper by outlining a promising direction for solving the reformulated problem.

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 in St. Louis
St. Louis, MO 63130-4899, USA
{*gorinsky,anshul,jst*}*@arl.wustl.edu*

Abstract

In this paper, we revisit the question of how much buffer an IP router should allocate for its output link. For a long time, the intuitive answer of setting the buffer size to the bitrate-delay product has been widely regarded as reasonable. Recent studies of interaction between queueing at IP routers and TCP congestion control proposed alternative answers. First, we expose and explain contradictions between existing guidelines for link buffer sizing. Then, we argue that the problem of link buffer sizing needs a different formulation. In particular, the chosen buffer size should accommodate not only common versions of TCP but also UDP traffic. Besides, our new formulation of the problem contains an explicit constraint of not engaging IP routers in any additional signaling. We conclude the paper by outlining a promising direction for solving the reformulated problem.

1. Introduction

Simplicity of IP (Internet Protocol) [19] is often cited as the primary reason for the tremendous 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 reduce their communication rates upon inferring congestion from implicit signs such as datagram loss. Many applications rely on TCP (Transmission Control Protocol) [21] for congestion control. Other applications communicate over UDP

(User Datagram Protocol) [20] and need to control congestion on their own; the choice of UDP often stems from dissatisfaction with TCP properties such as high variability of communication rates on short timescales or extra delay caused by retransmission-based in-order delivery.

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 sharing a bottleneck link, the total communication rate of the connections does not converge to a single value but oscillates within a range [7]. 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; for example, loss-driven congestion control can keep a big portion of a Droptail FIFO (First-In First-Out) buffer permanently occupied and thereby delay all forwarded datagrams.

The question of how much buffer an IP router should allocate for its output link has been traditionally formulated as a problem of fully utilizing the link without excessive queueing when the link becomes a bottleneck. For a long time, a widely accepted solution was to set the buffer size to the product of the link bitrate and round-trip propagation delay of a TCP connection. The guideline is easily derivable from a model where the TCP connection transfers a long file, is the only source of traffic on the bottleneck link, and halves its load on the link whenever it overflows the Droptail FIFO buffer. More elaborate recent analyses of models where multiple TCP connections share a bottleneck link indicate that the traditional rule of thumb has to be adjusted. Since different TCP connections may reduce their loads asynchronously, one guideline proposes decreasing the buffer size as the number of connections increases [3]. Another guideline suggests that the optimal buffer size grows proportionally to the number of TCP connections [17].

In this paper, we explore and explain contradictions between existing guidelines for link buffer sizing. After tracing the contradictions to differences in formulations of the link buffer sizing problem, we derive a consistent guideline from a more comprehensive model that reconciles contradictory assumptions made by the existing solutions. Then, we argue that even the reconciled model is too limited in scope. In particular, it is imperative that link buffer sizing should accommodate not only common versions of TCP but also UDP traffic. We propose a new formulation of the problem that incorporates this requirement. The new formulation also contains an explicit constraint of not engaging IP routers in any additional signaling. The constraint ensures that proposed solutions can be implemented in practice without undermining the lightweight design of IP networks. The paper also outlines a promising direction for solving the reformulated problem of link buffer sizing.

The rest of the paper is structured as follows. Section 2 reviews existing guidelines for link buffer sizing. In Section 3, we expose, explain, and reconcile contradictions between the existing guidelines. This section also shows that even the reconciled model is insufficient. Then, Section 4 provides a new formulation for the problem of link buffer sizing and discusses a promising avenue for solving it. Finally, Section 5 summarizes contributions of the paper.

2. Existing Guidelines for Link Buffer Sizing

It is the most appropriate to start our analysis of guidelines for link buffer sizing at the traditional rule of thumb that prescribes setting the buffer size to the **bitrate-delay product**. Although the guideline is often attributed to a 1994 paper by Villamizar and Song [26], there exist much earlier

references to this rule. For example, Jacobson's 1990 description of TCP Reno clearly identifies the bitrate-delay product as a condition for full utilization of a bottleneck link [14].

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 [7]. The assumed additive-increase multiplicative-decrease (AIMD) pattern approximates well the congestion-avoidance mode in such versions of TCP as Reno [14] and NewReno [11]. 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 simple model above considers only one TCP connection. How much buffer should a bottleneck link have if it serves multiple TCP connections? Recent studies of this question offer two mutually contradictory guidelines: over-square-root and connection-proportional allocation.

Over-square-root generalizes the traditional rule of thumb by prescribing to set the link buffer size to the bitrate-delay product divided by $\sqrt{n}$ where n is the number of TCP connections [3]. The guideline is derived from evidence that not all the connections lose 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, 24]. 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 link buffer size should be proportional to the number of TCP connections [17]. This alternative guideline stems from an observation that throughput of a TCP connection can suffer from a high loss rate and frequent retransmission timeouts if the link buffer does not accommodate at least few datagrams for each of the TCP connections.

3. Contradictions Reconciled

Although over-square-root and connection-proportional allocation agree that the traditional rule of thumb is inadequate for situations with multiple TCP connections, the new guidelines conflict on whether the buffer size should be decreased or increased. Furthermore, whereas over-square-root is just a refinement of the bitrate-delay-product rule, connection-proportional allocation makes no reference to either the link bitrate or network propagation delays. Which of the guidelines is correct? What are the reasons for the contradictions? We answer these questions in Sections 3.1 and 3.2 respectively. After tracing the contradictions to differences in assumptions made by the guidelines, Section 3.3 derives a consistent guideline from a more comprehensive model that reconciles the contradictory assumptions.

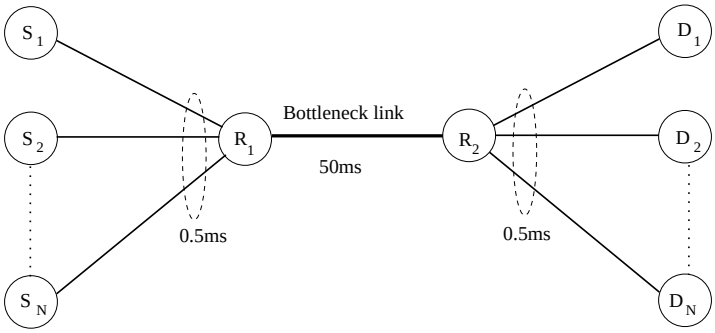


Figure 1: Traditional single-bottleneck network topology.

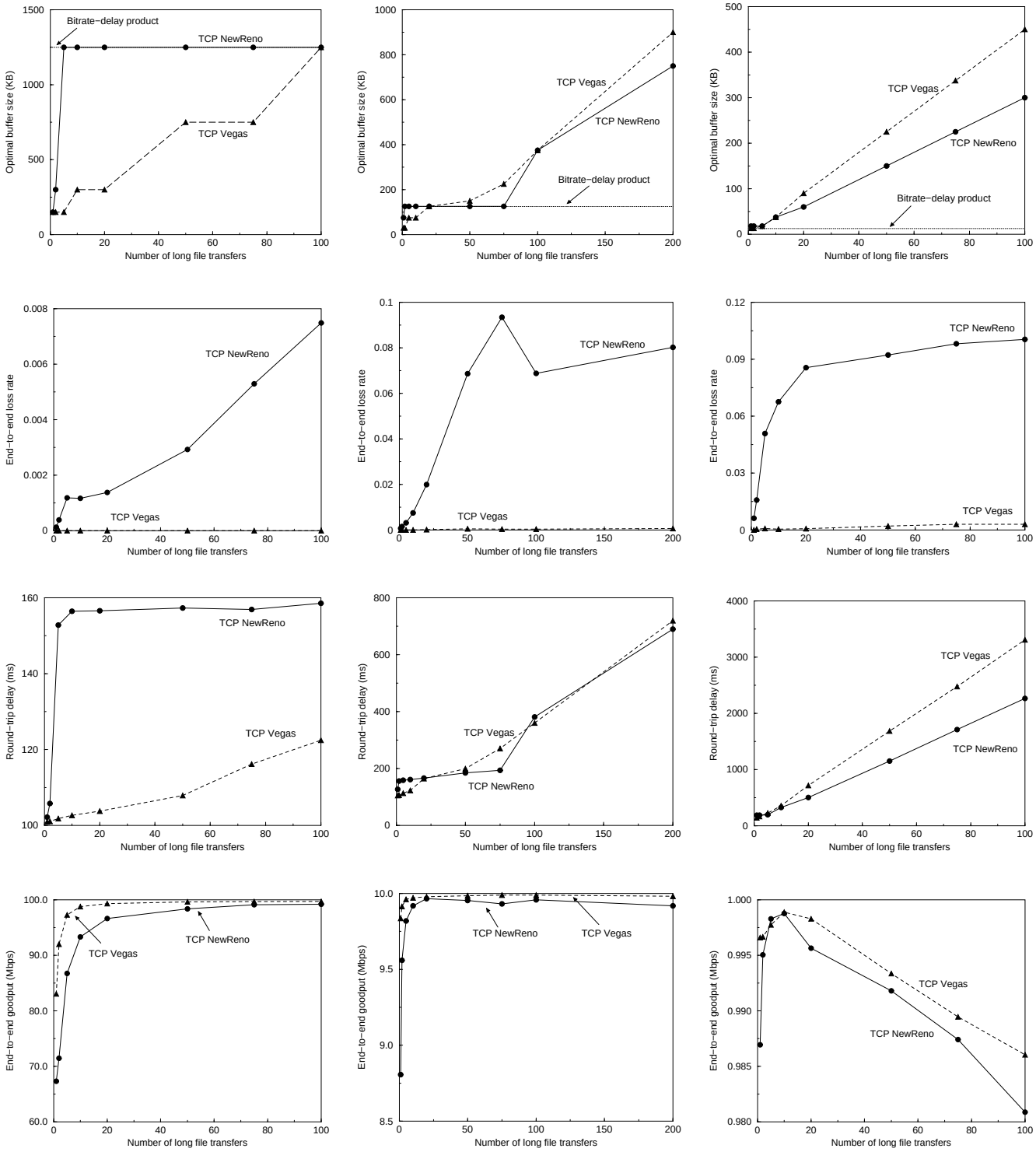
3.1. Experimental Validation

Since practice is the best judge of theory, we evaluate validity of over-square-root and connection-proportional allocation by conducting ns-2 [18] simulations in settings close to the assumed by the guidelines. Figure 1 shows the traditional network topology with a single bottleneck link in the middle. End systems S_i and D_i host respectively the sending and receiving ends of unicast applications. The value of i varies from 1 to N; hence, 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. The bottleneck 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 examine three values of the bottleneck bitrate: 100 Mbps, 10 Mbps, and 1 Mbps.

For a file transfer application, full utilization of the bottleneck link is not a goal in itself. What is important for the application is the amount of time it takes for the network to deliver the complete file from the sender to the receiver. Hence, the buffer size of the bottleneck link is optimal for a file transfer if the buffer supports the largest end-to-end goodput where end-to-end goodput is defined as the ratio of the file size to the file delivery time. In addition to end-to-end goodput, the end systems track loss rates and round-trip times to shed more light on queueing at the bottleneck link. We measure all the parameters over the whole experiment duration of 200 seconds.

Figure 2 presents our results. For the bottleneck bitrate of 100 Mbps, the original rule of thumb approximates the optimal buffer size precisely. The top graph in 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 the connection-proportional allocation guideline. The bottleneck bitrate of 1 Mbps provides even stronger support for connection-proportional allocation. The top graph in 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. Note that neither of the experiments validates the over-square-root guideline that prescribes a smaller buffer size for a larger number of long file transfers.



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

Figure 2: Experimental validation of the existing guidelines for link buffer sizing.

3.2. Analysis of Observed Results

First of all, what is wrong with the over-square-root guideline? Why does our evaluation show no reasons for decreasing the buffer size as the number of TCP connections grows? Appenzeller, Keslassy, and McKeown derive the guideline for a backbone link that serves tens of thousands connections. The proposers of over-square-root also indicate that the lack of reduction synchronization appears and reveals the $\sqrt{n}$ formula only when n is at least on the order of thousand connections [3]. Hence, to check the validity of the guideline for backbone links, the ranges explored in Figure 2 should be extended to include thousands of connections. Simulating such large numbers of long file transfers over a high-speed link was a feat that we were not able to achieve in ns-2. Furthermore, the situation 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 that serve fewer concurrent connections.

Whereas our results provide no sound basis for judging correctness of the over-square-root guideline for a backbone bottleneck link, Figure 2 shows clearly that over-square-root 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 our way, let us now consider 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 the connection-proportional allocation guideline. What is the reason for such dichotomy? What makes the ratio an important parameter? Our analysis confirms correctness of the logical derivations that led to the discoveries of both guidelines. The contradictory conclusions are due to differences in the assumptions made by the guidelines. The original rule of thumb assumes that a TCP connection operates uninterruptedly in the congestion-avoidance mode. This assumption can be wrong if the bottleneck link buffer does not accommodate at least few datagrams for each TCP connection. Instead, the connections experience high loss rates that lead to frequent retransmission timeouts and large reductions in communication rates. On the other hand, the guideline of connection-proportional allocation strives to keep each TCP connection in the congestion-avoidance mode but does not concern itself with utilizing the bottleneck link completely. Hence, if the ratio of the bitrate-delay product to the number of file transfers is high, connection-proportional allocation yields underutilization of the link and thereby suboptimal goodput of the file transfers.

3.3. Reconciling Guideline

Section 3.2 traced the contradictions between the existing guidelines for link buffer sizing to assumptions made by the guidelines. Now, we construct a consistent guideline from a more comprehensive model that reconciles the contradictory assumptions. The model combines the requirement of keeping each TCP connection in the congestion-avoidance mode with the goal of utilizing the bottleneck link completely. Then, a simple derivation leads us to the following reconciling guideline:

set the link buffer size to the maximum of the bitrate-delay product and connection-proportional buffer size.

Is the constructed guideline *the* final word in link buffer sizing? Did the previous paragraph just solve the link buffer sizing problem for good? Even our own answer to the above questions is not affirmative. First, the new guideline does not reflect the asynchrony of individual load reductions in scenarios where the number of TCP connections is at least on the order of thousands. This particular complication is not significant in our opinion. One can amend the reconciling guideline by suggesting that in situations with so many TCP connections, the link buffer size should be *set to the maximum of the values prescribed by over-square-root and connection-proportional allocation*. However, there exist more important reasons to doubt the overall approach behind the new guideline as well as the guidelines it reconciles.

Figure 2 reports the optimal buffer size not only for NewReno but also for Vegas version of TCP [6]. With TCP Vegas, the optimal buffer size does not depend on the bitrate-delay product at all and always remains approximately proportional to the number of connections. Vegas is different from Reno and NewReno in two respects. First, Vegas uses delay variations as an implicit sign of congestion and can curb transmission without overflowing the bottleneck link buffer. Second, Vegas uses an additive-increase additive-decrease algorithm in the congestion-avoidance mode. The second difference implies that the amplitude of the total load oscillations under TCP Vegas is proportional to the number of connections. Consequently, the optimal buffer size is also proportional to the number of TCP Vegas connections. This conclusion is interesting because it shows that the reconciling guideline fails to offer a tight upper bound on the buffer requirements of TCP Vegas. More importantly, the conclusion highlights the limited scope of our problem formulation: *Find the size of the Droptail FIFO buffer that optimizes performance for a specific application of long file transfer over a common version of TCP*. We believe that the problem of link buffer sizing should be reformulated in a more general context; in particular, the assumed model should account for needs of different applications.

4. New Problem Formulation for Link Buffer Sizing

At first glance, the task of constructing a general but solvable formulation for the link buffer sizing problem seems daunting. In the most general case, the correct answer depends on too many factors such as policies used for link scheduling [8, 13] and datagram discard [12, 15, 25]. We do not attempt to tackle the most general case; we still consider only Droptail FIFO buffers, which are the most typical in the modern Internet. Instead, we focus our attention on diversity of Internet applications in Section 4.1. Then, Section 4.2 enhances the new formulation of the link buffer sizing problem with an explicit constraint of not engaging IP routers in any additional signaling.

4.1. Accounting for Diverse Needs of Internet Applications

Transferring a long file over TCP is a common application in the Internet. It is perhaps also the application where TCP congestion control is the most relevant for the problem of link buffer sizing. Since queueing delay at the bottleneck link is substantially smaller than time to deliver a long file,

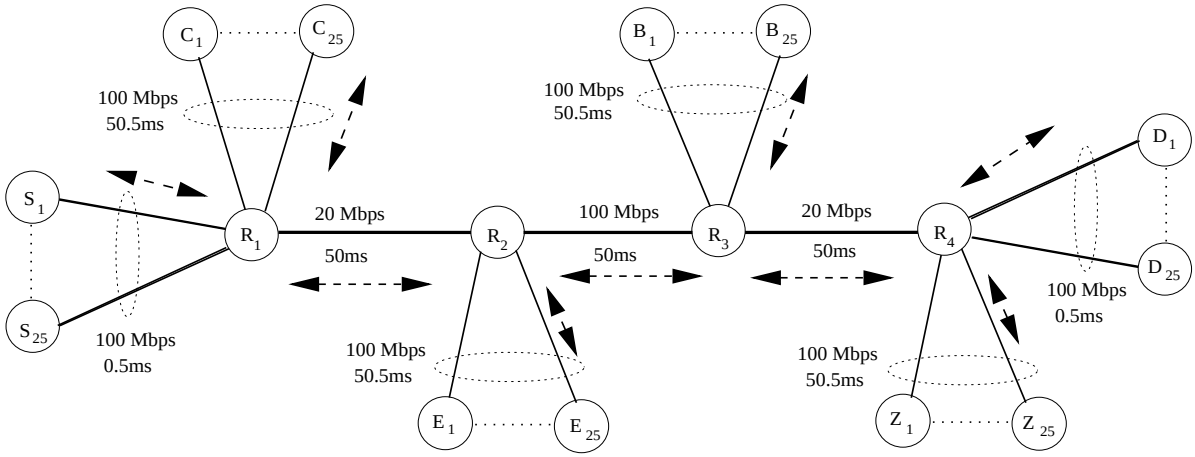


Figure 3: Multiple-bottleneck network topology.

utilization of the bottleneck link becomes the primary factor affecting the end-to-end goodput of long file transfers. This is the reason why the link buffer sizing problem is traditionally formulated as a problem of fully utilizing the bottleneck link without unnecessary queueing.

Figure 2c clearly demonstrates that setting the buffer size to the value optimal for long file transfers can result in long queueing delay on the order of seconds. However, extensive queueing can have a devastating effect on short file transfers over TCP and interactive streaming over UDP. Hence, it is imperative that the reformulated problem of link buffer sizing addresses needs of these other important types of Internet applications.

The optimal trade-off between the bottleneck link utilization and queueing delay is different for short file transfers over TCP. It has been shown that a small constant buffer size is sufficient to optimize performance of the short transfers regardless of their number [2, 3, 4]. Interactive streaming puts even more emphasis on small delays. In particular, delivering datagrams within round-trip time of few hundred milliseconds becomes essential because larger delays make the interaction uncomfortable for human perception.

To assess whether and how a single buffer size can reconcile the divergent optimization objectives characteristic for different types of Internet applications, we conduct ns-2 experiments in the multiple-bottleneck topology depicted in Figure 3. The topology corresponds to a common Internet scenario where backbone links are utilized lightly, and access links constitute bottlenecks. 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. The value of i varies from 1 to 25. The following applications generate traffic in 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. The applications that rely on TCP employ either NewReno or Vegas versions of it. Each of the links uses FIFO Droptail buffering. Note that all the

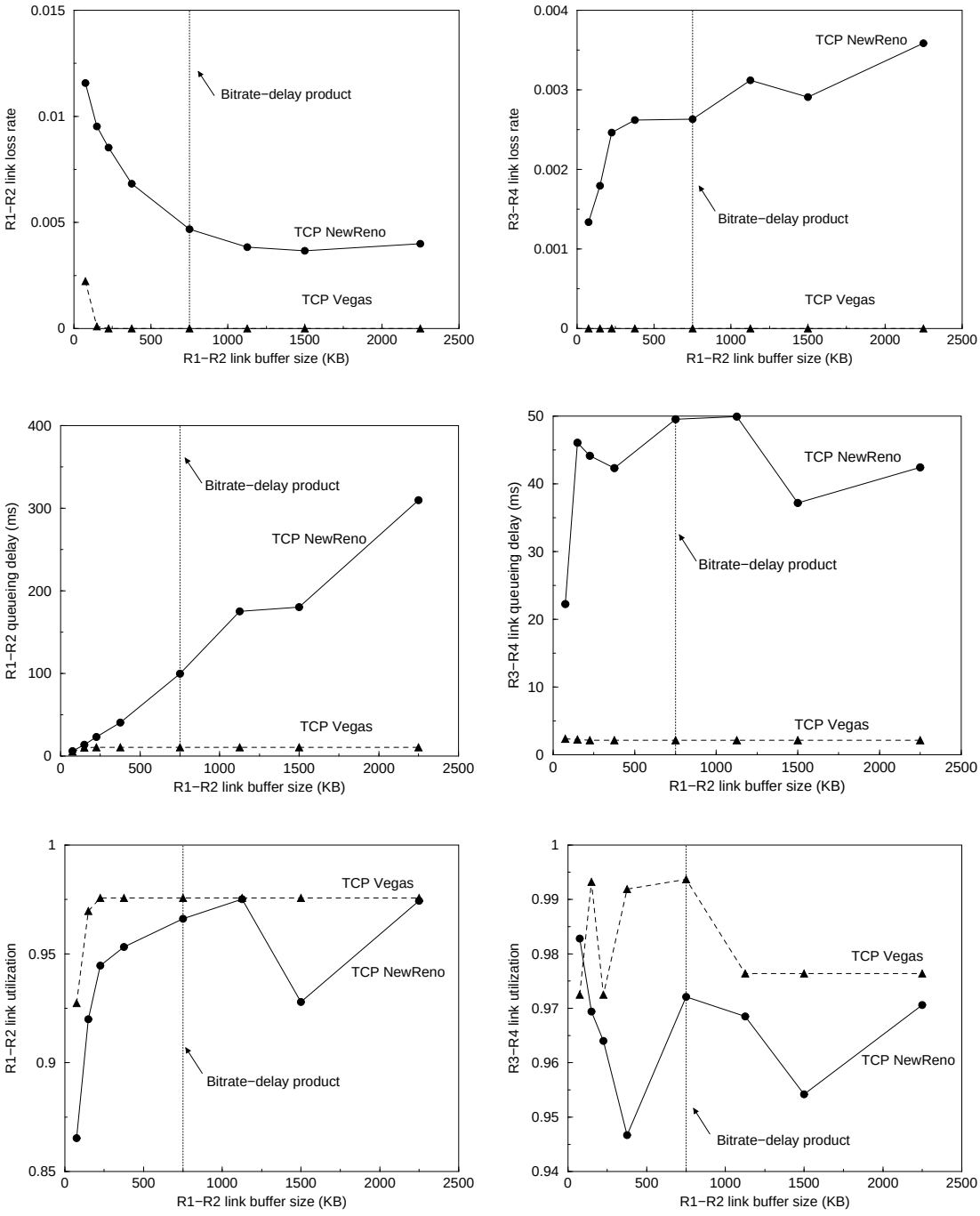
applications have the same round-trip propagation delay. Hence, the bitrate-delay product for a link is the same for every application using this link. Our evaluation focuses on the applications transmitting data from end systems S_i. Links R_1-R_2, R_3-R_4, R_4-R_3, and R_2-R_1 are the four bottlenecks 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 the results measured for links R_1-R_2 and R_3-R_4 when the buffer size for link R_1-R_2 varies but the buffer sizes for 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 applications: round-trip time for the interactive video, goodput for the long file transfers, and delivery time for the 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 recommended setting inflates the round-trip time for the interactive video above 600 ms, a value that is already unacceptably large for the application.

The above experiments demonstrated that choosing a large Droptail FIFO buffer to accommodate long file transfers over TCP can dramatically hurt interactive streaming over UDP. Is it possible to reconcile the needs of both application types by selecting a smaller buffer? Previous studies provide evidence supporting the affirmative answer to this question; in particular, it has been shown that selecting a buffer size that is suboptimal for file transfers does not cause a substantial degradation in the average goodput, even though the individual goodputs of the file transfers can become more variable [4, 22]. To examine the sensitivity of end-to-end goodput to suboptimal buffer sizes, we repeat the Section 3.1 experiments in settings where the buffer size for the 10 Mbps bottleneck link is set 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, setting the buffer size for the bottleneck link significantly below the optimal value can provide long file transfers with end-to-end performance that is reasonably good.

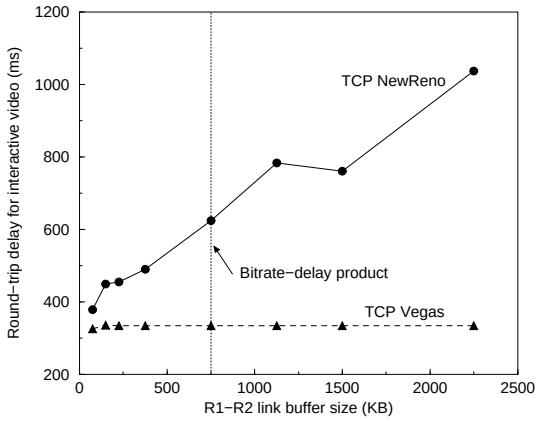
A simple analysis for the worst-case scenario with a single connection confirms this conclusion for such common TCP versions as Reno and NewReno. Even with a minimal buffer, the load of the single TCP connection in the congestion-avoidance mode oscillates between 50% and 100% of the full link utilization. Hence, the congestion avoidance still delivers the long file at a rate that is at least 75% of the bottleneck link bitrate. The 25% decrease in end-to-end goodput of long file transfers seems to be a reasonable price for making viable those UDP applications that cannot tolerate long queueing at the bottleneck buffer configured according to the existing guidelines. However, if the 25% penalty deems unacceptable for long file transfers, the penalty can be substantially reduced without increasing the size of the bottleneck link buffer. Instead, long file transfers can reduce the penalty by adopting smoother congestion control protocols such as TCP Vegas or application-level protocols for smooth data delivery over UDP [5, 10, 16]. For example, in spite of the minimal buffer, a simple change of the AIMD decrease factor from 0.5 to 0.875 [9, 23] raises the bottleneck link utilization from 75% to 94%.

In this section, we presented compelling arguments that the problem of link buffer sizing should account for needs of diverse Internet applications. The arguments lead us to the following new formulation of the problem:

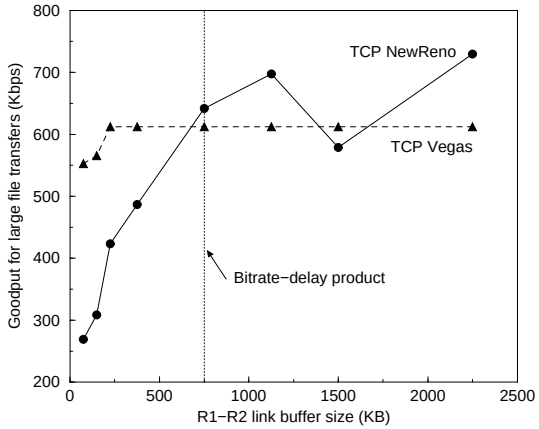


(a) Performance of link R1-R2 (b) Performance of link R3-R4

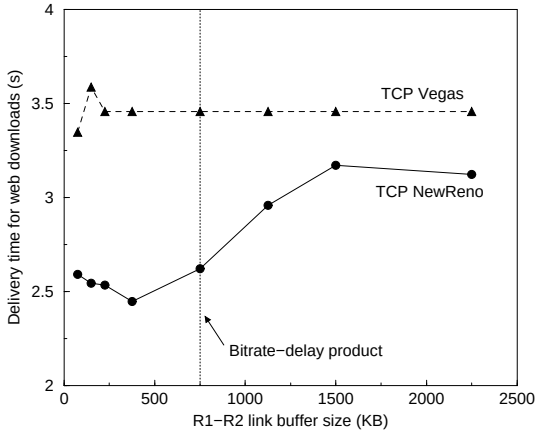
Figure 4: Performance of individual links in the multiple-bottleneck topology.



(a) Round-trip time for interactive video



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



(c) Delivery time for short web downloads

Figure 5: End-to-end performance for different types of Internet applications.

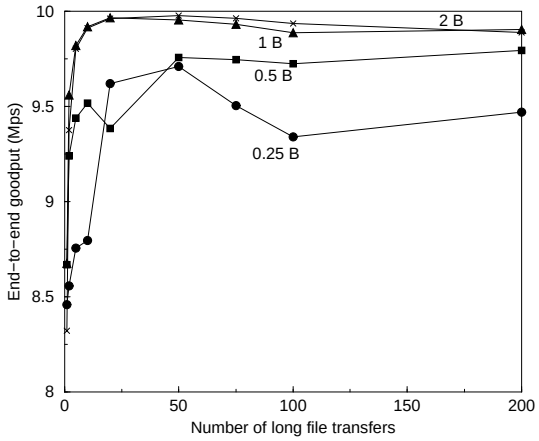


Figure 6: Impact of suboptimal buffer sizes on end-to-end goodput of file transfers over TCP NewReno.

Find the link buffer size that accommodates both TCP and UDP traffic.

Our discussion also indicated that the reformulated problem has solutions that can satisfy all major types of Internet applications. The next section enhances the problem formulation with a constraint on acceptable solutions.

4.2. Explicit Constraint of No Additional Signaling

In our opinion, it is imperative that a proposed rule for link buffer sizing can be implemented in practice without undermining the lightweight design of IP networks. To reflect this requirement, we enhance the problem formulation with an explicit constraint of not engaging IP routers in any additional signaling. Hence, our final formulation for the problem of link buffer sizing becomes as follows:

Find the link buffer size that accommodates both TCP and UDP traffic.
An acceptable solution cannot engage IP routers in any additional signaling.

Whereas the extra property may seem not only reasonable but also obvious for a good solution, satisfying this requirement is problematic even for the existing guidelines. For example, some of the guidelines require the router to know the number of long file transfers but IP does not provide the router with reliable means to acquire this knowledge. There exist well-studied techniques for grouping received datagrams into flows according to IP addresses, port numbers, and other datagram header fields. However, using the flow statistics to estimate accurately the number of long file transfers over a specific version of TCP appears to be difficult [17].

The guidelines that involve the bitrate-delay product are even more difficult to implement because the IP router has no knowledge of round-trip propagation delays for passing traffic. It seems infeasible to detect the round-trip propagation delays by simply monitoring the headers of received

datagrams. Replacing the round-trip propagation delay in a rule for link buffer sizing with round-trip time (RTT which includes propagation and queueing) is an adjustment that can facilitate implementing the rule without any additional signaling. However, the following simple analysis shows the danger of the adjusted rule. Consider a scenario where a bottleneck link has a Droptail FIFO buffer and bitrate B. The link serves a single connection that adheres to a common TCP version such as NewReno or Reno. The round-trip propagation delay for the connection equals D. The connection delivers a long file and stabilizes in the congestion-avoidance mode. The router manages to infer the average RTT of the connection precisely and sets the link buffer size to the product of B and the inferred RTT. Initially, the router sets the buffer size to the optimal value of BD. The load of the sender on the network oscillates between BD and $2BD$, and the queueing delay oscillates between 0 and D. Since the average RTT is $1.5D$ at this point, the router sets the buffer size to $1.5BD$. The load of the sender on the network oscillates now between $1.25BD$ and $2.5BD$, and the queueing delay oscillates between $0.25D$ and $1.5D$. Since the average RTT is $1.875D$ at this point, the router sets the buffer size to $1.875BD$, and the vicious circle of the harmful unnecessary increases in the link buffer size and queueing delay continues.

The illustrated danger of replacing the round-trip propagation delay in a guideline for link buffer sizing with RTT is not limited to settings where routers update link buffer sizes automatically. Similar scenarios can occur – albeit on much longer timescales – if human operators follow the approximated guideline to increase the link buffer sizes manually.

Since exact and safe implementation of the existing guidelines appears to be difficult, it is reasonable to inquire about current practices for configuring link buffers. According to anecdotal evidence, it is common for router operators to allocate all the buffering memory provided by the router 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. Unfortunately, choosing such large buffers can lead to excessive queueing. End-to-end Internet measurements provide indirect confirmation of the aforesaid practices and their consequences: it has been shown that RTT within the *same* TCP connection can vary by several seconds [1].

Given the discussed difficulties with implementing the existing guidelines, is it possible at all to find an acceptable solution for the reformulated problem of link buffer sizing? In Section 4.1, we concluded that small link buffers can accommodate well both TCP and UDP traffic. We also believe that a small link buffer represents a solution implementable without engaging the IP router in any additional signaling. The small buffer size can be chosen based on the link bitrate and other information that is readily available locally.

5. Conclusion

This paper revisited the old question of how much buffer an IP router should allocate for its output link. For a long time, the intuitive answer of setting the buffer size to the bitrate-delay product has been widely regarded as reasonable. Recent studies of interaction between queueing at IP routers and TCP congestion control proposed alternative rules. In this paper, we exposed, explained, and reconciled contradictions between the existing guidelines for link buffer sizing. We argued that the problem of link buffer sizing needs to be redefined in a more general context. Then, we proposed such a new problem formulation: *Find the link buffer size that accommodates both TCP*

and UDP traffic; an acceptable solution cannot engage IP routers in any additional signaling. Our experimental studies and theoretical analyses revealed a promising direction for solving the reformulated problem: set the link buffer size to a small value based only on local information, e.g., the link bitrate.

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.E. 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.E. 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] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker. Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms. In *Proceedings ACM SIGCOMM 2001*, August 2001.

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

[7] 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.

[8] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *Proceedings ACM SIGCOMM 1989*, September 1989.

[9] S. Floyd, M. Handley, and J. Padhye. A Comparison of Equation-Based and AIMD Congestion Control. http://www.aciri.org/tfrc/, May 2000.

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

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

[12] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[13] P. Goyal, H. M. Vin, and H. Cheng. Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks. In *Proceedings ACM SIGCOMM 1996*, August 1996.

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

[15] D. Lin and R. Morris. Dynamics of Random Early Detection. In *ACM SIGCOMM 1997*, September 1997.

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

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

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

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

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

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

[22] L. Qiu, Y. Zhang, and S. Keshav. Understanding the Performance of Many TCP Flows. *Computer Networks*, 37(3-4):277–306, 1999.

[23] K.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.

[24] 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.

[25] B. Suter, T.V. Lakshman, D. Stiliadis, and A.K. Choudhury. Buffer Management Schemes for Supporting TCP in Gigabit Routers with Per-Flow Queueing. *IEEE Journal on Selected Areas in Communications*, 17(6):1159–1169, June 1999.

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