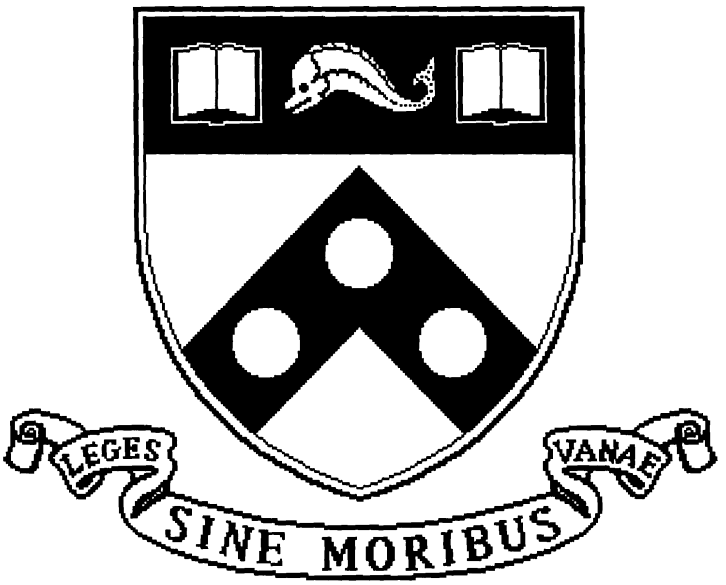
Congestion control by bandwidth-delay tradeoff in very high speed networks: the case of window-based control

MS-CIS-94-55
DISTRIBUTED SYSTEMS LAB 80

Hyogon Kim
David J. Farber



University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

October 1994

Congestion control by bandwidth-delay tradeoff in very high-speed networks: the case of window-based control

Hyogon Kim, David J. Farber
Distributed Systems Laboratory

October 24, 1994

Abstract

Increasing bandwidth-delay product of high-speed wide-area networks is well-known to make conventional dynamic traffic control schemes "sluggish". Still, most existing schemes employ dynamic control, among which TCP and ATM Forum's rate-based flow control are prominent examples. So far, little has been investigated as to how the existing schemes will scale as bandwidth further increases up to gigabit speed and beyond. Our investigation in this paper is the first to show that dynamic control has a severe scalability problem with bandwidth increase, and to propose an entirely new approach to traffic control that overcomes the scalability problem. The essence of our approach is in exercising control in bandwidth domain rather than time domain, in order to avoid time delay in control. This requires more bandwidth than the timed counterpart, but achieves a much faster control. Furthermore, the bandwidth requirement is not excessively large because the bandwidth used would have been idled if dynamic control were used. Effectively, we trade bandwidth for smaller control delay and we call our approach Bandwidth-Latency Tradeoff(BLT). While the control in existing schemes are bound to delay, BLT is bound to bandwidth. As a fallout, BLT scales tied to bandwidth increase, rather than increasingly deteriorate as conventional schemes. Surprisingly, our approach begins to pay off much earlier than expected, even from a point where bandwidth-delay product is not so large. For instance, in a roughly AURORA-sized network, BLT far outperforms TCP on a shared $150Mbps$ link, where the bandwidth-delay product is around $60KB$. In the other extreme where bandwidth-delay product is large, BLT outperforms TCP by as much as twenty times in terms of network power in a gigabit nationwide network. More importantly, BLT is designed to continue to scale with bandwidth increase and the performance gap is expected to widen further.

1 Introduction

With the onset of high-speed wide-area networks, a significant portion of ongoing research is being devoted to efficiently support nonreal-time data traffic. To mention a few, TCP running on ATM network testbeds exposed some problems yet to be solved in interfacing with ATM[RF94, oAml94], ATM/AAL-level congestion control mechanisms such as link-by-link flow control[KC94] and rate-based traffic management[YH94] have been proposed to control ABR traffic congestion, and switch algorithms have been investigated to support transport layer data traffics over ATM

networks[FJ93, Flo94]. One big open question in supporting data traffic on very high-speed networks, however, is if conventional traffic control mechanisms can fully exploit large bandwidth provided by underlying high-speed fabric, such as ATM networks. Leaving other problems, such as how to implement TCP/IP on ATM standards/implementations, to be addressed by others[oAml94, RF94], we concentrate on a more fundamental aspect of the problem: whether the conventional way of controlling source rate can still be effective over upcoming larger bandwidth-delay product networks. Indeed, whether window-based or rate-based, and whether on transport layer or on lower(*i.e.*, ATM/AAL) layer, recently proposed methods all seem to adopt dynamic source rate control[Jac88, KBC94, YH94, HKM93, RJ90]. Although it is well-known by now that large bandwidth-delay products make traditional dynamic rate control "sluggish"[Kle92], there has been little effort to investigate existing methods' scalability with ever increasing bandwidth-delay products. A few previous works address this problem only limitedly, *e.g.* mainly for LAN environment[KC94], or with infinite source assumption[KMT93], or in networks with relatively small bandwidth[MK92]. Therefore, a more generalized and in-depth investigation on this issue is needed. Moreover, recent researches show bursty nature of data traffic is not mitigated by traffic aggregation. In fact, aggregating bursty data streams typically intensifies it, instead of smoothing it[LTWW93, GW94]. So future high-speed networks will see intensified burstiness contributed by many bursty traffic sources. Also, the burstiness will exist at every time scale, from milliseconds to minutes to hours, so load fluctuations that are shorter than round trip time will be frequent. This is bad news to conventional feedback(dynamic) traffic control because no feedback control can deal with a congestion that is shorter than its feedback loop[Jai90]. Conversely, the conventional schemes will not be able to exploit temporarily available network bandwidth left by load fluctuation, where even a relatively short fluctuation will amount to large data size(that could have been transported) in large bandwidth-delay product network, again causing throughput to fall.

In this paper, as a first step, we investigate the effect of large bandwidth-delay product on dynamic window protocols such as TCP, show the scalability problem, and then attempt to develop a new approach to overcome such problem. We take an entirely different and novel approach to the traffic control to cope with large bandwidth-delay product with better scalability. Increases in the bandwidth will logically lengthen the feedback loop because while more data is transmitted during a given time interval, control speed cannot be accelerated due to speed-of-light delay. Slow feedback control will get even slower with further bandwidth increase. Although bandwidth increase creates this problematic trend, if there is a way to appropriate this bandwidth back to quicken the control, we would have more resource to mobilize to solve the problem. Moreover, we would not need to worry about scalability of such approach because with bandwidth increase, it would also scale. The main purpose of this paper is to show that this is feasible. To make a long story short, we propose to do dynamic control in *bandwidth domain*, rather than *time domain*. We will be thereby trading bandwidth for smaller control delay, hence obtaining hopefully better performance. So our approach is called Bandwidth-Latency Tradeoff(BLT; pronounced "Blitz"). The details of how this

approach can be materialized in the context of window control is discussed in section 3, but first, to state our goals of this investigation:

1. to develop a traffic control methodology to overcome large bandwidth-delay product
2. to establish the relationship between
   - bandwidth
   - propagation latency
   - transported data size
   - traffic model in high-speed network, *e.g.*, self-similarity, *etc.*

   to determine the range where the above methodology is feasible

The organization of this paper is as follows: Section 2 briefly surveys a few typical window-based congestion control schemes. We give the conceptual framework of bandwidth-latency tradeoff in section 3. In section 4 we show simulation results comparing window-based bandwidth-latency tradeoff technique and ordinary TCP. Section 5 summarizes our discussion and lists future works.

2 Related work

In this section, we briefly survey dynamic traffic control algorithms with emphasis on window-based schemes.

2.1 DECbit

DECbit[RJ90] is one of the earlier congestion avoidance scheme that require cooperation of network switches and traffic sources. In the DECbit binary feedback scheme, the congested network switches set congestion indication bit in the network layer header of a data packet, where the congestion is indicated by the average queue length greater than or equal to 1. Non-congested switches ignore the congestion indication bit field. At the destination, the bit is copied into the transport header of the acknowledgment packet which is then transmitted to the source. The source updates its window once every two round-trip times, and if at least 50% of the bits examined are set during the interval, the window size is reduced from its current value W_c to $\beta \times W_c$. Otherwise it is increased to $W_c + \alpha$, where $\beta = 0.875$ and $\alpha = 1$. The scheme uses minimal amount of feedback from the network, with just one bit in the network layer header to indicate congestion. The scheme addresses the important issue of fairness and achieves the goal to some extent.

When there is a transient change in the network, DECbit is designed to adapt to the change and converge to the efficient operating point. But in [KKM92], the authors show that DECbit is quite ignorant of short transient load fluctuations in the network unless the change lasts a long time. Also, it is shown that DECbit's additive window increase scheme is sometimes too conservative. Its

slow linear increase causes the file transfer time to far exceed those of all other schemes compared in [KKM92], including TCP.

2.2 TCP congestion control and avoidance algorithms

For congestion control in TCP, a series of successful optimizations on congestion detection and avoidance algorithms have made TCP increasingly more efficient compared with other window-based methods such as DECbit. Jacobson and Karels[Jac88] introduced Slow Start and Congestion Avoidance algorithms to TCP. When a connection begins, or a timeout occurs for a data segment, TCP starts to reopen congestion window from 1 segment until it hits half of the previous window size that caused the loss of the segment and subsequently the timeout. Then a separate algorithm, Congestion Avoidance, takes over to increase the window size roughly by 1 segment per roundtrip time. Albeit cautious, Slow Start exponentially increases the window size, while Congestion Avoidance increases it linearly. By probing the next packet loss point slowly, Congestion Avoidance attempts to maximize the inter-loss interval, thereby improving throughput. If only Slow Start is used, frequent packet loss will lower the throughput. In the Tahoe version of TCP(4.3-tahoe BSD TCP)[Jac88, Ste94], another algorithm, Fast Retransmit, is also included to discover packet loss without waiting for timeout of the lost packet. If three duplicate ACK packets are received, the source decides that a packet has been dropped. The source uses both Fast Retransmit and retransmission timers to detect packet losses. In Reno version of TCP, Fast Recovery algorithm[Jac90] is added. With the Fast Recovery algorithm, the TCP source does not slow-start after inferring (by Fast Retransmit) that a packet has been dropped. Instead, the TCP source effectively waits for half round trip time and halves the congestion window. The source retransmits the dropped packet and uses incoming duplicate ACKs to clock outgoing packets. TCP "Vegas"[BOP94] further refines TCP in three points. First, it improves upon retransmission algorithms. It uses time stamp to get more accurate RTT estimate, and does not necessarily wait until 3 duplicate ACKs arrive to begin retransmission. Second, it exercises a certain proactive congestion avoidance measure. Vegas keeps track of expected throughput and actual sending rate, and compares them as it tries to control the sending rate by linearly changing window size. Third, Vegas modifies Slow Start to detect congestion during the Slow Start phase. To detect and avoid congestion during slow-start, Vegas allows exponential growth only every other RTT. When the actual rate falls below the expected rate by a certain amount, Vegas changes from slow-start mode to linear increase/decrease mode.

A recent proposal incorporates explicit congestion notification(ECN) in TCP[Flo94, FJ93], so that TCP is informed of incipient congestion without dropping any packet. There have been a range of rate-based congestion control schemes for ATM layer or higher layer with ECN[FJ93, RJ90, New94], and this scheme assumes such information is available from lower layers. The advantages of ECN approach for TCP are first, unnecessary packet drops can be avoided, and second, sources can be informed of congestion fairly quickly. The disadvantages are the absence of ECN field in IP packet headers and the uncertainty of the delivery of the ECN if the ACK packet

with the ECN message is itself dropped by the network. In LAN environment, TCP with ECN showed higher throughput for bulk data transfer, and low delay for delay-sensitive data traffic such as telnet. But for wide-area networks, the simulations suggest that ECN does not give significant advantages. This is because the TCP clock granularity is less likely to be a problem in wide-area network time scale, and similarly, the additional delay of waiting for a retransmit timer timeout has a less significant impact on performance. However, as network speed increases and TCP overhead reduces, this scheme could be potentially rewarding.

In high-speed network environment where propagation latency is large, we expect that not only additive-increase/multiplicative decrease can be too slow but even Slow Start can also be slow. Although Slow Start opens window exponentially, when the maximum window size is large due to large bandwidth-delay product, it still takes significant number of round trip times to open up a large enough window. In high-speed networks, the available bandwidth keeps growing, and the size of a significant number of files will be on the order of a few roundtrip time $\times$ bandwidth. Our experience with Internet shows that "small" files are the major source of data traffic[CDJM91]. In this sense, one RTT increase per one segment window size increase in TCP Vegas may not be desirable especially in high-speed networks.

Another possible source of problem is in the timer granularity. When queueing delay is decreasing and the roundtrip delay is less than a few tens of milliseconds, an average of 250 milliseconds for the retransmission timer timeout to expire is a long time, for similar reasons discussed in the above paragraph.

And finally, as with all other window-based control schemes, if load fluctuation is shorter than RTT, which is intensified under self-similarity assumption and bandwidth increase, TCP may be unable to react timely to such fluctuations.

2.3 Dynamic window control in ATM networks

So far we have looked at transport level congestion control algorithms. For completeness, we will look at window control algorithms for newly emerging ATM networks.

Hahne et al.[HKM93] envision a wide-area ATM network that offers a sliding window-controlled service for data. In their scheme, end-to-end cell window size is dynamically changed at intervals as short as one round-trip time. The window size change request affects the buffer size reserved for the particular connection at the network switches. They showed that the window protocol and higher layer protocols such as TCP work well together.

The above work resembles a credit-based link-by-link flow control[KBC94], except that it is an end-to-end rather than a link-by-link scheme. The link-by-link scheme is Flow-Controlled Virtual Circuit(FCVC) and it is one of the proposals for ATM flow control[KC94]. The adaptive credit allocation mechanism is one variation of the original FCVC proposal that dynamically changes switch buffer allocation according to a connection's usage of reserved bandwidth. The original credit-based flow control method is a sliding window mechanism, and the adaptive allocation algorithm allows

the window size to change dynamically during the conversation. It is needed not only to reduce buffer requirement of the original scheme, but also to cope with data traffic that cannot be easily pre-characterized with parameters[New94], where the characterization is a requirement to allocate enough buffer that guarantees the absence of packet loss.

Although not window-based, there are a few rate-based proposals for ATM flow control[YH94, New93][1]. One notable thing about these rate-based flow control schemes is that they also adopt additive-increase/multiplicative-decrease dynamic rate change that strongly reminds us of window size changing mechanisms of transport layer protocols like TCP and DECbit. In this paper, we limit our discussion to window protocols, postponing the discussion on how to apply our approach to rate-based protocols for a future paper. We also omit the discussion on the ATM level flow controls. Although the implications of our discussions are not limited to any one particular layer of protocol stack, we concentrate on transport layer traffic control in this paper. We believe that it is time to look into the scalability of existing transport layer traffic control schemes especially when there are active ongoing researches on how to implement existing transport layer protocols such as TCP on the emerging high-speed ATM networks[RF94, BD94]. This is with the expectation that TCP will remain as the prime transport protocol for the future data traffics, and it will be very hard to deploy any pure ATM network flow control in apparently heterogeneous network of the future[New94, Cro94].

3 Concept of bandwidth-latency tradeoff

3.1 Problem definition

The reason that dynamic congestion/flow control will be inadequate in very high bandwidth-latency product network is twofold. With increasing bandwidth and constant propagation delay, the logical feedback control loop keeps lengthening[Tou93]. This (physics) is the first reason why dynamic feedback congestion control will not scale, *i.e.*, the control is bound to the delay rather than the bandwidth. With this irreversible trend the feedback controls will continue to lag, and may not be able to react to the network condition timely. One thing that would make control lag worse is self-similarity, which means we should expect intensified burstiness at all time scale[LTWW93, GW94]. Namely, in our high-speed wide area backbone, the load will fluctuate fast and violently even at small time scale. The control depending on slow feedback simply cannot adjust to, or exploit, such changes.

The second reason is in the cautious nature of dynamic controls. Most window control schemes and some rate control schemes follow conservative rate changing principles. DECbit uses additive-increase/multiplicative-decrease[RJ90], and so does rate-based ATM traffic control[YH94, MK92], and TCP[Jac88] linearly increases window size in Congestion Avoidance phase while exponentially decreasing it upon packet loss. The reason behind this conservativeness is the fact that the average

[1] Recently ATM Forum voted to support rate-based flow control among many proposals[New94].

queue length at the bottleneck switch begins to increase exponentially upon congestion, and the system will stabilize only if the traffic sources throttle back at least as quickly as the queues are growing[Jac88]. Since symmetric multiplicative increase will cause wild oscillation and poor throughput, the additive increase is used. Although safe, additive increase can be too conservative to exploit large bandwidth that dynamically becomes available. For instance, in [MK92], DECbit is shown to perform poorly due to this. TCP showed better performance than DECbit, owing to the exponential increase element, Slow Start. Slow Start exponentially opens a congestion window just after packet loss is indicated by retransmission timer timeout, or at the beginning of a connection. Although exponential increase is fast enough for small propagation delay networks, the initial part of the increase can still be slow under large propagation delay, especially considering large bandwidth and finite data size being transported. Some rate-based schemes also use a certain form of "slow-start"[KKM92].

In essence, conventional traffic control schemes have inherent limitations that do not keep up with bandwidth increase. We aim to show how this limitations are manifested, and quantify the boundary where the limitations call for totally different way to control traffic. We also intend to discuss a straw-man design of the new approach, and compare its performance with that of a conventional scheme, TCP. But first, we discuss a methodology to overcome the effect of large bandwidth-delay product in this section.

3.2 Conceptual solution: control in bandwidth domain

We will later show that delay will limit the usefulness of conventional feedback controls via experiment in section 4, but until then let us just suppose it *is* the problem. With this assumption, we search for a methodology to overcome the curse of propagation delay which is relatively increasing as bandwidth increases.

Obviously the simpliest (but *unreal*) solution to eliminate the feedback delay would be to "move" the control point(*i.e.* congestion control algorithm at the source) to where the control action actually applies (*i.e.* the congested switch), eliminating the physical distance barrier. In other words, the traffic source closer to the congested switch will be able to react to the congestion faster than a source father located. Rather than working on an elaborate control mechanism to compensate for the effect of propagation delay, which existing schemes attempt to do, our approach advocates this approach of moving the control point to the controlee, but only *virtually.*

In our *bandwidth-latency tradeoff* (BLT; pronounced "Blitz") approach[2], a data stream to the network carries *multiple* performance parameter values at the same time. To create the parameter space, the source uses the bandwidth otherwise wasted unused, due to the inherent conservativeness of dynamic control methods as described in Section 3.1. The controlee(switch/destination) can then

[2]BLT is totally different from NETBLT[CLZ87], a bulk data transfer protocol. Although BLT and NETBLT share the common philosophy that dynamic window change is too slow to achieve high throughput, NETBLT advocated rate-based flow control and became one of the progenitors of the current rate-based protocols which exercise more sophisticated rate control. Application of BLT to rate-based schemes will be discussed in future work.

choose an appropriate parameter value that exactly fits the network condition at the time of its control action. Since the source yields the right to exercise the control action to the controlee, the control point is effectively "moved" to where the controlee is. This is only rational because it is the controlee that knows exactly which control action should be taken at any moment. Now that the control action takes place immediately at the controlee, this approach does not suffer from long propagation delay, but at the expense of bandwidth used in creating the multiple parameter value space, hence there is a bandwidth-latency tradeoff.

To a window-based congestion control, the 'performance parameter' is congestion window size. BLT dictates that a data stream carries multiple number of windows with different sizes. Then network switches(possibly including the receiving end of the connection) determine which window size is acceptable given the current network load. This saves time for a congested switch to communicate with the traffic source, so faster control can be achieved. In particular, adaptive rate change algorithms such as Congestion Avoidance and Slow Start become unnecessary. In BLT, the source does not need to blindly "feel for" a maximum acceptable window size over time because a data stream provides entire menu of window sizes or source rates to the switch simultaneously and the switch simply chooses one. This approach gives results as if the source transmitted with precisely the largest window size that the switch can accept on the first try.

3.3 A window-based incarnation of BLT

In this section, we design a window-based congestion control scheme based upon the bandwidth-latency tradeoff approach. The design guidelines are as follows:

- By overcoming the inherent problem of dynamic feedback control, the new scheme should improve performance over conventional schemes. And it should *scale* better with bandwidth increase. Also, in a small bandwidth-delay product environment its performance should not be worse than in conventional schemes.
- It should not cause bias or unfairness between connections or between different traffic characteristics any more than conventional methods.

3.3.1 Data stream structuring

With the above guidelines, we begin with how the source structures a data stream. Recollect that the essence of BLT idea is to provide the entire spectrum of window sizes to the network. So the only constraint in structuring a BLT data stream is that it should contain multiple number of windows with different sizes. We plan to investigate the performance implications of different window sizing schemes, but in this paper, we decided to use exponential increase in the window sizes, *i.e.*,

$$\begin{aligned} |W_0| &= 1 \\ |W_i| &= \gamma \cdot |W_{i-1}|, \quad i \geq 1 \end{aligned}$$

where $|W_k|$ is the size of k^{th} window in packets and γ is the increase ratio. We set $\gamma = 2$, again deferring the effect of different values of γ until future work. And of course, we could use additive increase so that,

$$|W_i| \quad = \quad \alpha + |W_{i-1}|, \quad i \geq 1$$

The exponential increase in BLT should not be considered to be emulating Slow Start in TCP, and as briefly mentioned in Section 3.2, additive-increase/multiplicative decrease is also irrelevant here because window size probing is not necessary in BLT. Window sizing in BLT is just a way for the source to structure a data stream into different sized windows so that network switches can select one appropriate size among them, and there is no reason that BLT should emulate any of the dynamic window resizing algorithms.

If p is the first packet in the newly opened portion of the congestion window, we will have the windows with exponentially increasing sizes under the current structuring scheme as shown in Figure 1 where the file is q in packets. Conceptually, the sender sends the entire (remaining) file

$$\begin{array}{lcl}
W_0 & : & (p) \\
W_1 & : & (p,\ p+1) \\
W_2 & : & (p,\ p+1,\ p+2,\ p+3) \\
\vdots & : & \vdots \\
W_k & : & (p,\ p+1,\ \ldots,\ p+2^k-1) \\
\vdots & : & \vdots \\
W_n & : & (p,\ p+1,\ p+2,\ \ldots,\ q)
\end{array}$$

Figure 1: Conceptual structure of a BLT stream carrying $q - p + 1$ packets in n different size windows, $\gamma = 2$

at once. All windows $W_0..W_n$ are sent out at the same time to the network so that the network switch can pick one of the window sizes to pass. However, it is not clear as shown in Figure 1 as to how to schedule packet transmissions of each window in reality. Moreover, there are many redundant transmissions of the same packets. For example, packet p is transmitted $n + 1$ times in the above example. Finally, it would make the switch simpler if the switch does not need to actively decide which window size to pick among different window sizes, since this will typically take computations based on (average) queue length, number of active connections, *etc.* So, if we restructure and order the windows as presented in Figure 2 and let W'_i be transmitted followed by W'_{i+1}, the scheduling problem is simplified and the redundant transmission problem vanishes. The most interesting aspect in the above implementation is that while redundant transmission problem is eliminated, *virtually* we still send out the same entire window spectrum to the network that

$$
\begin{array}{lcl}
W'_0 & : & (p) \\
W'_1 & : & (p+1,\ p+2) \\
W'_2 & : & (p+3,\ p+4,\ p+5,\ p+6) \\
\vdots & : & \vdots \\
W'_k & : & (p+2^k-1,\ p+2^k,\ \ldots,\ p+2^{k+1}-2) \\
\vdots & : & \vdots \\
W'_m & : & (p+2^{k+1}-1,\ p+2^{k+1},\ \ldots,\ q)
\end{array}
$$

Figure 2: Actual structure of the BLT stream

was enumerated in Figure 1. This is because when the switch takes in a series of windows, *e.g.* $\boldsymbol{W}_k = (W'_0,\ W'_1,\ \ldots,\ W'_k)$, this set of consecutive windows will approximate one of the original windows, *i.e.*,

$$|\boldsymbol{W}_k| = |W_{k+1}| - 1$$

This results as if the switch *selected* W_{k+1} (except packet $p+2^{k+2}-2$) as the window size that it can *maximally* take in from the original window space given in Figure 1. Note than any window sizes shown in Figure 1 can be approximated this way. By making the switch absorb the maximal number of packets that the switch can digest, we can exploit the dynamically available bandwidth left by fluctuating load. But note that the last part of $\boldsymbol{W}_k$(or, approximately, W_{k+1}) can be cut short due to queue overflow and some other reasons that will be discussed later. The acceptable number of consecutive windows, k, is *implicitly* determined by the queue size, without involving computations and associated data structures at the switch. One small advantage of using exponentially increasing window size is that even with a small number of bits, we can encode a large number of packets. With π bits, we can give priorities to 2^π windows, which in exponential scheme is $\sum_{i=0}^{2^\pi-1} 2^i$ packets. With only $\pi = 4$, we can encode $2^{16} - 1 = 64K$ packets.

No matter how clever the ordering scheme is, if the consecutive window $\boldsymbol{W}_k$ is transmitted in a burst without any information on the structure of the data stream, the stream will overflow an unsuspecting switch and block other connections from entering the switch buffer. Therefore, we need a mechanism to signal the degree of opportunism of the packets in a stream. In our ordering schemes, k value denotes "warped" round trip times. In TCP, W_k would be sent after $k-1$ round-trips at its earliest. But in BLT it is transmitted in the first stream(burst), $k-2$ round trip times earlier[3]. So the larger the k, the more opportunistic the packets in W_k are. One mechanism to discriminate against more opportunistic packets is to give lower priority to such packets. We simply consider k as the *priority* of W_k, with larger k representing lower priority. In other words,

[3]This is a bad example. Here BLT is not emulating TCP Slow Start. The exponential increase of window sizes is only a coincidence.

the priority simply encodes how much earlier the particular window is being transported than its time. In Figure 3, we show how TCP congestion window size varies over time with the example of Slow Start and how BLT windows are transmitted over time.

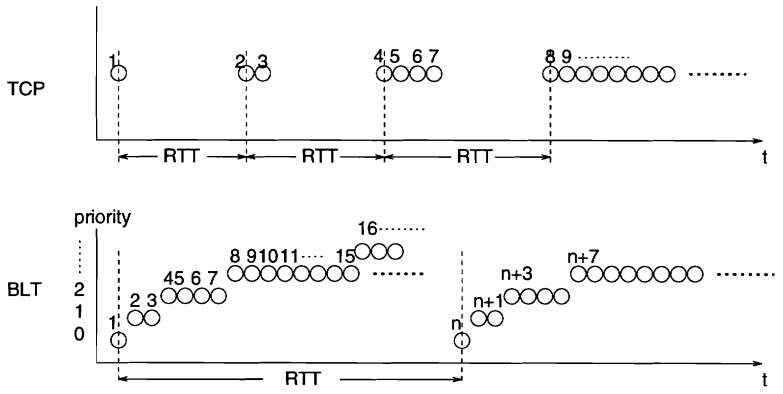


Figure 3: Comparison of TCP(in Slow Start) and BLT stream structures

3.3.2 Switch queueing discipline

Once the data streams have this structure, we need a priority discarding discipline at network switches to impose higher dropping probability on more opportunistic packets. In this paper, we assume that the switches work with the following algorithm:

1. `if` $\neg full(Q)$, `then` use FIFO discipline regardless of priorities.
2. `else` /* full */
   (a) `if` $\forall j\ P(pkt) \geq P(Q[j])$, `then` drop pkt.
   (b) `else`
      i. Find $max(j)$ such that $P(pkt) < P(Q[j])$.
      ii. `for` $k = j + 1$ `to` $QSIZE$
      $Q[k-1] \leftarrow Q[k]$ /* advance */
      iii. $Q[QSIZE] \leftarrow pkt$ /* insert at the end */

where pkt is the incoming packet, Q is the switch queue, and P is the priority function. This algorithm is not the only possible instance that fits the BLT idea, but we use the above algorithm in this paper. The algorithm is very similar to the model analyzed in [PF90, THP93]. In fact, the algorithm has some room for improvement. For instance, when the queue is full the first packet from the end that has a lower priority, rather than the lowest priority packet in the entire queue, is dropped. For fairness' sake, the algorithm should have dropped the lowest priority packet. We will correct this feature in our future work. From the algorithm description, we can easily notice that

longer bursts will be punished when a new stream comes on. This is only fair, and gives shorter, interactive traffic priority over non-interactive bulk transfer type traffic.

In terms of implementation, the above algorithm is much more complex than simple FIFO, but one approach that allows for the implementation of the policy considered in this paper is discussed in [Cha91]. We assume in this paper that the above algorithm can be implemented in hardware in the network switches. We are currently investigating if BLT can be applied to other switching disciplines. We believe priority FIFO scheduling is not the essential part of BLT scheme, and BLT could be applicable to other switching disciplines that exercise a certain control action for congestion control.

3.3.3 Priority reset

In the current version of our scheme, a source transmits all the remaining data in one single burst. We assume the same cumulative acknowledgement(which is used by TCP) is used at the receiving end. When a packet n is dropped at one of the network switches, the loss will be signaled to the sender explicitly via duplicate ACKs in case out-of-order packets arrive at the destination, or implicitly via timeout[4]. When the packet drop is signaled, whether or not the source is still transmitting the stream n belongs to, the source starts another burst (quitting the old one in case it *was* transmitting) starting from n. However, this time, it gives priority 0 to packet n. In fact, the entire priority space is shifted over the sequence number space when ACKs arrive. So packet $n+1$ and $n+2$ gets priority 1, and so forth(Figure 3).

Since entire files can be transmitted in one burst, it may seem that our approach will aggravate congestion in the network. It is true that our BLT sources generate large bursts especially in the earlier part of the transmission. However, in our experiment shown in Section 4, we will see that the opposite is true. Namely, we observed lower average utilization at the bottleneck switch with BLT scheme than with TCP. This can be explained by two reasons. First, BLT sources finish transmissions quickly. So on the average, fewer BLT streams linger in the network than TCP, with the same arrival rate of the connections. With more active connections in the systems, TCP faces higher probability of packet drops and associated timeouts, Slow Starts, shorter Congestion Avoidance phases, and so forth. This again leads to more new connections coming into the system before old connections finish their transmissions, sustaining high level of utilization. Second, all the BLT sources get a relatively fair chance at the switches owing to the prioritizing schemes. One long transmission cannot occupy the switch queue, which leads to higher average response time and hence higher utilization as explained in the first reason.

Although we only consider file-transfer like traffic in this paper, we can easily extend the priority reset scheme to interactive traffic. Whenever a user submits new data, we treat it as a new burst. So we give the packets in the new burst starting from priority 0. But we do not reset priority for any data remaining in the transmission buffer at the sender, submitted prior to the current data.

[4] We use the same acknowledgement scheme as TCP just for fair comparison.

The remaining data in the transmission buffer is only reset when packet loss is signaled as discussed above. This way, short interactive traffic gets higher priority over bulk data transfer, withing the same priority reset scheme designed above.

In this paper we only intend to test the feasibility of BLT approach, so we do not attempt to optimize the way BLT transmits. But in future version we will investigate how to avoid this obvious overhead.

3.3.4 Underlying assumptions

BLT needs more assumptions than TCP to work. First, all the sources need to follow the priority encoding scheme described above. Otherwise, one abusive source marking all its packets with highest priority will deceive network switches to pass its packets before others. One separate but related problem with this is how to prioritize other types of traffic. To real-time traffic we can simply give the highest priority as many approaches propose[KC94], but it is rather a tricky problem to decide which priority to give to non flow-controlled traffic, such as UDP. We could give lower priority to UDP-like traffics, but how low is one question. The problem is more difficult if we give the same treatment to UDP-like traffics as flow-controlled traffics. The second assumption is regarding the switches. For BLT to work, every multiplexing point along the path of a connection should discard packets based on priority. The multiplexing points may work in multiple layers of the protocol hierarchy. If BLT works on a higher layer (although BLT concept is not limited to a particular layer), then the priority information should filter down to lower layer and the multiplexing point working on the lower layer should discard its data PDUs based on the priority. If any one of the multiplexing point does not exercise priority discarding along the path of the connection, then it will soon overflow and BLT will not work. One counter-argument is that connection multiplexing/demultiplexing is disadvantageous because multiplexed streams must be treated identically. Feldmeier[Fel93] argues that multiplexing interferes with our ability to provide only those services that are required, making high-speed protocol configuration difficult. It is also shown that less multiplexing/demultiplexing decreases processing cost while increasing memory cost. However, it is beyond the scope of this paper to address all these issues, and we only investigate the feasibility of BLT concept itself, assuming all the requirements are met and BLT is ready to work. And we will examine if BLT can outperform conventional schemes under such favorable assumptions.

Finally, although not a necessary assumption for BLT, we assume throughout the paper that the sender is not the bottleneck. In other words, the sender has enough bandwidth to drive the network with BLT streams. The bottleneck is assumed to exist in the network. This is realistic because the bandwidth of one high-performance workstation is comparable to a highspeed network bandwidth as of today.

4 Simulation

In this section, we investigate the feasibility of BLT approach by comparing its performance with that of TCP. The TCP we use in this experiment implements all the congestion control mechanisms used by Reno TCP, namely Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery. Also it has some Vegas-like features, which are time-stamping for more accurate estimation of RTT and immediate retransmission upon the arrival of 1 duplicate ACK[5].

One difference between our TCP emulation and other implementations is that our TCP is packet-oriented rather than byte-oriented. Also we assume the packet size to be fixed. But such packet-oriented emulations of TCP have been extensively used in other research works without noticeable side-effects[RF94, MK92]. As to BLT, we use the window-based BLT discussed in Section 3 without modification.

4.1 Simulation setting

The simulation setting in this investigation is intentionally simplified and generalized to clearly observe the effects of a few important parameters, which could get diffused in a more complex environment. For instance, in configuring the simulated network we use only two parameters associated with bandwidth-delay product, *i.e.*, bandwidth and propagation delay. And in terms of workload, we also use only two parameters: mean arrival rate of connections and data size to transfer. The network model is one of the simplest, with one bottleneck link as shown in Figure 4, which is conceptually similar to what is used in [FRW92]. Now we give the detailed description on our simulation setting beginning with the basic assumptions to be used throughout the entire experiments.

4.1.1 Some assumptions

As we discussed in Section 3, we assume that traffic sources have enough bandwidth to inject BLT streams into the network, and only consider the case that network is bottleneck. So, for example, we do not consider such case that multiple BLT source processes on the same host congesting a network interface, making the interface device a bottleneck[6].

We assume that a receiver has infinite buffer space, and therefore only the congestion window size matters. So our TCP emulation does not have the problem of insufficient receiver window size to "fill the pipe". This makes sense because real TCPs also provide Window Scale option[JBB92] for the same purpose.

In this experiment we limit our traffic characteristics to ftp-like bulk data transfer although future investigations will generalize it to include interactive and intermittent traffic(*e.g.*, as in

[5]In our simulation, out-of-order packet delivery is not allowed, so 1 duplicate ACK is a non-ambiguous sign that a packet has been dropped.

[6]Multiple bottleneck case will be investigated in future work.

[HKM93]). But by tuning file size parameter to have very small value, we expect to get some insight on how the congestion control schemes will behave for interactive traffics. In fact, [CDJM91] shows that 80% of all interactive conversations send as much data as the aberage bulk transfer conversation – although conversely it means that bulk transfer applications send a smaller amount of data than is often assumed.

We only model transmission delay and propagation delay. Other delays such as operating system overhead, TCP overhead, hardware interface overhead are not modeled. This is to expose the inherent limit of the traffic control schemes more clearly. Considering the continuing improvement on processing power and the large magnitude of propagation delay compared with the other delays, we do not believe this assumption poses any significant problem on the implications of the experimental results obtained through this investigation.

Figure 4 shows our model network. There are unspecified number of traffic sources whose

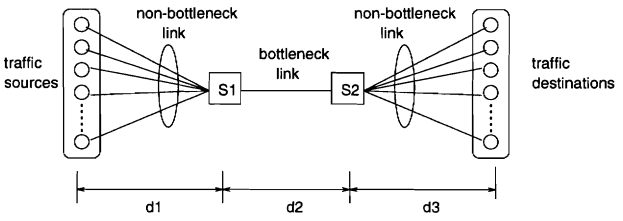


Figure 4: Configuration of the simulated network

aggregate arrival rate to the network is determined by mean arrival rate parameter. There is only one bottleneck link, which is between switches $S1$ and $S2$[7]. Twice the sum of the link propagation delays(d_1, d_2, d_3) determines propagation RTT. In this experiment we set $d_1 = d_2 = d_3$ without loss of generality. The switches have one (link bandwidth $\times$ link RTT) worth of buffer space, and all connections share the buffer. The packet size used is $500B$, which is close to Internet packet size mean[CDJM91]. Finally, we assume that ACK packets cannot be dropped at the switches. But in reality, ACK packets can be bundled, or even dropped when congestion occurs. In future investigation, we will loosen the assumption and allow ACK packets to be dropped.

4.1.2 Network configurations

To see the effect of bandwidth-delay product on the performance of congestion control schemes, we vary two parameters *network bandwidth* BW_N and *propagation delay* D. We determine network configuration by the two parameters without changing the network topology given in Figure 4. Generally, we expect to see the following spectrum of configurations according to given BW_N and D:

- high bandwidth-delay product network:

[7]Switch $S2$ is redundant as far as the experiments in this paper are concerned.

  - high BW_N, large D

- medium bandwidth-delay product network:
  - high BW_N, small D or,
  - low BW_N, large D

- low bandwidth-delay product network:
  - low BW_N, small D

Specifically, we use nationwide network size of $RTT = 30ms$ and AURORA testbed size of $RTT = 3ms$ as D values. For BW_N, we use $150Mbps$, and $1Gbps$, with the bottleneck bandwidth BW_b ranging from $\frac{1}{5}$ to $\frac{3}{5}$ to full BW_N. Varying BW_N from $150Mbps$ to $1Gbps$, we intend to observe the scalability of the experimented schemes with increasing bandwidth. Also by varying BW_b, we intend to observe the effect of $\frac{BW_b}{BW_N}$ on the performance scalability.

4.1.3 Workload

To control the load on the network, we vary the *arrival rate* λ of connections. The connections arrive with the inter-arrival time having exponential distribution, and generate file transfer-like traffic. The major reason that we do not consider interactive traffic in this paper is that such traffic carries so little data that more or less it is unaffected by flow control. For instance, Caceres *et. al.* showed that about 90% of interactive conversations send less than $10KB$ bytes over a duration of 1.5 to 50 minutes. Therefore interactive traffic is an inappropriate traffic class to use in an investigation of flow/congestion control schemes. To get a good confidence interval quickly, we made all connections send the same sized file. The file size F in each simulation varies from $10KB$ to $100KB$ to $1MB$. Previous research on Internet traffic reveals that 75-90% of the conversations belonging to bulk transfer applications send less than $10KB$ of data. Moreover, almost all bulk transfer size is less than $1MB$. So our assumption on the file size faithfully follows the observation[CDJM91].

By varying λ and F, we expect the following load spectrum:

- high load:
  - large F with large λ

- medium load:
  - small F with large λ or,
  - large F with small λ

- low load:
  - small F with small λ.

For each network configuration and each file size , we vary λ until the utilization at the bottleneck node, ρ, approaches near 100%.

4.1.4 Performance metrics

Average response time R is the average of all $(T_{f,k} - T_{i,k})$, where $T_{f,k}$ is the finished time and $T_{i,k}$ is the arrival time of the k^{th} connection to the system. Namely,

$$R = \frac{\sum_k (T_{f,k} - T_{i,k})}{k}$$

Response time has been used in [MK92] as the primary performance measure, easily quantifying the relative performance of a flow or congestion control scheme. Throughput Θ is the ratio of the number of finished connections to the total time elapsed. Or, it is the ratio of transferred bytes to the total time elapsed. In this paper, we adopt the second convention, because we need to unify the measure when F varies. So,

$$\Theta = \frac{\sum_{k,t=0}^{t=T} |c_k|}{T}$$

where $|c_k|$ is the file size in bytes that connection k carries and T is the total elapsed time to finish k connections. Once we get R and Θ, we can calculate *network power*[GHKP78]. The goals of maximizing throughput and minimizing response time are mutually contradictory in that all methods to increase throughput result in increased response time as well and vice versa[JR88]. Because it is clear that throughput and response time are really redundant metrics, they are combined into a single measure known as the network power, defined at any resource as[8]:

$$P = \frac{\Theta^\alpha}{R}$$

A higher power means either a higher throughput or a lower delay in a given system; in either case it is considered better than a lower power. We can uniquely determine the power characteristics for a given system as shown in Figure 5 by only varying the load(X axis). So we compare the performance of two different systems, one using TCP and the other using BLT, in terms of the power characteristics by giving the same system configuration parameter values and varying the load. In the above formula, when $\alpha = 1$,[9] the point at which the power is maximized is the "knee" of the delay curve, which is our desired operating point[RJ90]. Congestion avoidance schemes attempt to operate the network at the knee of the overall response time curve. At this operating point, the response time does not increased substantially because of queuing effects. Furthermore, the incremental throughput gained for applying additional load on the network is small. Since the two measures Θ and R compete with each other, P has a single maximum if the system is

[8] In this paper we abstract the whole network as a single resource and so the Θ and R above are system-wide throughput(total number of packets sent for all users divided by the total time) and system-wide response time(averaged over all users) giving us *system power*[JR88].

[9] By setting $\alpha > 1$, one can favor file traffic by emphasizing higher throughput. Similarly, by setting $\alpha < 1$ one can favor terminal traffic by emphasizing lower response time[JR88].

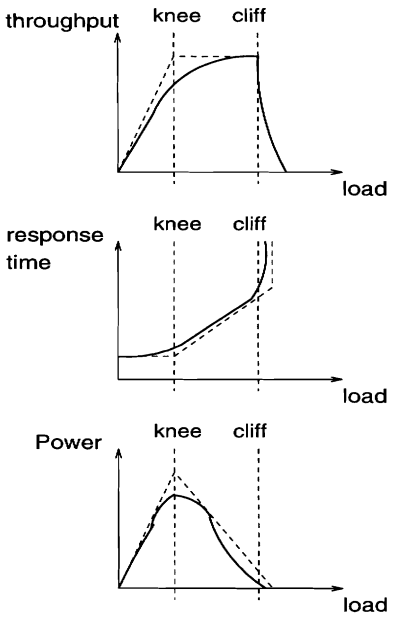


Figure 5: Network performance as a function of the load(from [JR88])

lossless[Kle78]. Our system *is* lossless[10], because a connection does not leave the network until it is finished. So for example if a system u has ten times the power of a system v(*i.e.* $P_u = 10 \cdot P_v$), it could be because $\Theta_u = 2 \cdot \Theta_v$ and $R_u = \frac{1}{5} R_v$, among numerous possibilities. Network power has been extensively used to quantify network performance[RJ90, Kle79].

Intuitively, power represents how far we can "push" the given network system to gain better performance, where the system is considered to consist of the physical bandwidth and the congestion control mechanism given in this paper. The increase of physical bandwidth will result in higher power anyway, but it is the efficiency of the congestion control mechanism that will determine the magnitude of power increase. A congestion control scheme that cannot effectively drive newly available network bandwidth will yield lower P than what can be achieved. In our paper, we use the network power as the primary performance measure that uniquely characterizes the performance of a congestion control scheme given the same raw bandwidth.

One might ask why the response time matters in $100KB$ data transfer because it should be a non-interactive bulk traffic. This, however, may not be true any more. Among bulk transfers, services like `gopher`, `www`, `wais` and other browsing/retrieval services are interactive, being classified as *interactive bulk* class[SCL93]. The data transfer size of one data item request in `www` for example, easily goes over a few tens of kilobytes. Researchers soon expect a widespread demand for an emerging generation of computer based applications that will expand such traffic class[SCL93]. In fact, the traffic of this type(especially `www`), absolutely exploded recently, in one observed site

[10] Loss should not be confused with packet drops. Loss occurs when a connection arrives to the system and leaves without being completed.

sustaining growth of 300-fold/year with no immediate signs of slowing down. And the volume of the traffic grew extremely rapidly by about a factor of 750 per year, even faster than the number of `www` connections. So the average volume of data of one `www` request is growing. At this pace, `www` traffic would surpass the volume of `ftpdata` traffic in four more months[Pax94]. So $100KB$ may well be interactive, let alone $10KB$. But $1MB$ should probably be regarded as bulk transfer. In this paper we do not vary α to address this issue, but this will be considered in our future work.

Another important performance criterion is *fairness* across all the users of the network. There are many definitions of fairness, but in our particular setting it is the equal distribution of bottleneck bandwidth across all outstanding connections. As we mentioned in the design guidelines in Section 3.3, fairness is one of the design goals of BLT. To compare the fairness of BLT and TCP, we adopt the *fairness function* f as defined in [JR88] as our metric:

$$f = \frac{(\sum_{i=1}^{n} \Theta_i)^2}{n \sum_{i=1}^{n} \Theta_i^2}$$

where Θ_i is i^{th} user's throughput and n is the number of independent users who are sharing the same resource. This function has the property that its value always lies between 0 and 1. For example, $f = 1$ for a maximally fair allocation but if only k of n users receive equal throughput and the remaining $n-k$ users receive zero throughput, the fairness function value is $\frac{k}{n}$[Jai91]. Although our main goal was to find a scheme that scales performance better with bandwidth increase, fairness is one thing we should not sacrifice. In our experiments, we not only compare the power of BLT and TCP, but we also show the fairness characteristics of the schemes.

Summarizing this section, we have the following parameters:

- network bandwidth (BW_N) and bottleneck bandwidth (BW_b)
- propagation delay (D)
- mean arrival rate of connections (λ)
- file size to transfer (F)

And as primary performance metrics, we will use the followings:

- network power ($P = \frac{\Theta}{R}$)
- fairness function (f)

4.2 Results and analyses

4.2.1 Under reference configuration and load

The reference network configuration is a $150Mbps$ network with $RTT = 30ms$, where each connection has $100KB$ to send. Namely,

- $BW_N = 150Mbps$
- $D = 30ms$
- $F = 100KB$

Figure 6 shows the average file transfer times of TCP and BLT with the reference parameters. X axis shows mean arrival time of connections (λ), and Y axis represents the file transfer time (R). Note that R is the system-wide average of file transfer times. The figure also shows the 90%

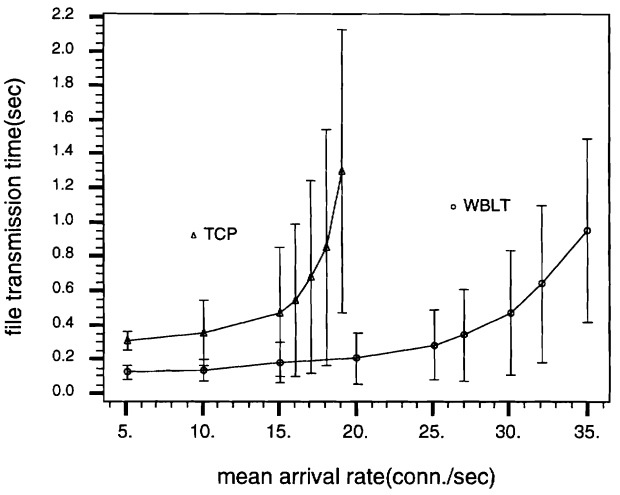


Figure 6: File transfer times of TCP and BLT as a function of λ in the reference network with $BW_b = 30Mbps$

confidence interval. We can see BLT consistently outperforms TCP with the speedup of more than 200% in R. Moreover, TCP diverges at $\lambda \approx 20conn./sec.$ while BLT does not until $35conn./sec.$ This is because the bottleneck switch $S1$ has a lower utilization with BLT than with TCP(Figure 7). This is seemingly counterintuitive because BLT transmits the entire remaining file on every burst. The reason that BLT keeps lower utilization is that there are a fewer number of connections in the system with BLT. As shown in Figure 6, BLT finishes transmission quickly. When we view the network as a whole to be a server, the jobs, *i.e.* the connections, departs the system earlier in BLT. By Little's Theorem we know that there will be more connections lingering in the system using TCP. This not only implies a larger average utilization at the bottleneck, but the probability that TCP connections will cause *global synchronization*[SZC91], will be larger. In fact, we observe this phenomenon in our simulations. Figure 8 shows the queue length variation in TCP and BLT in a randomly selected part of the simulation with the reference parameters. We clearly see that there are indications of global synchronization in TCP case while in BLT we cannot find any such

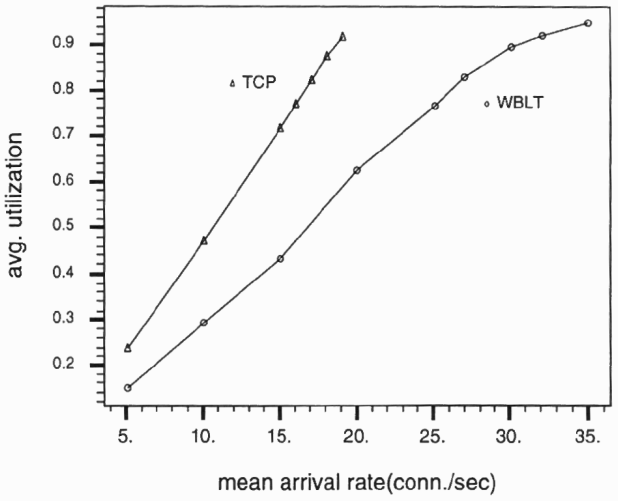


Figure 7: Bottleneck switch utilization in the reference network with $BW_b = 30Mbps$

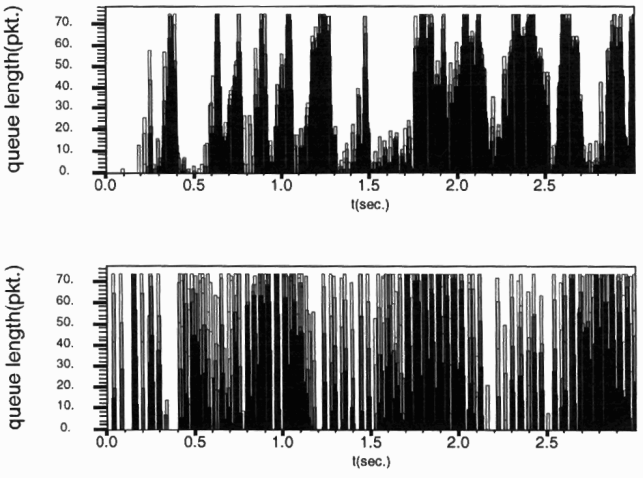


Figure 8: Instantaneous queue length change over time: TCP(up) and BLT(below)

tendency. Also, we can observe in the initial parts of the graphs that BLT quickly jumps up to exploit available bandwidth, but TCP takes time to warm up with Slow Start.

We can conjecture several reasons why BLT finishes transmission more quickly than BLT on the average. First, while BLT switching discipline prevents a burst from blocking other bursts entering the switch queue(Section 3), TCP switches can discriminate the packets that come after a large burst. This unfairness in TCP can increase the average response time R. Second, BLT has no cautious window size probing phases while TCP goes through various kinds of adjustment phases like Slow Start and Congestion Avoidance, which takes time. Finally, BLT provides the switch with more than sufficient data, and it can still pass many out-of-order packets even after packet drops for a particular connection. With reordering at the receiving end, this can sometimes contribute to a faster transmission.

When we vary the bottleneck bandwidth from $30Mbps$ to $90Mbps$ to $150Mbps$ we get the same results. Figure 9 shows the throughput of TCP and BLT with different bottleneck sizes. Roughly Θ increases proportionally to λ(X axis is in log scale), and BLT shows better throughput than TCP especially when λ is large. Note that higher utilization does not necessarily lead to higher throughput, which is obviously the case here. When we say "throughput", we will mean goodput in this paper. It is not shown where BLT throughput tops out in Figure 9 but we can see that

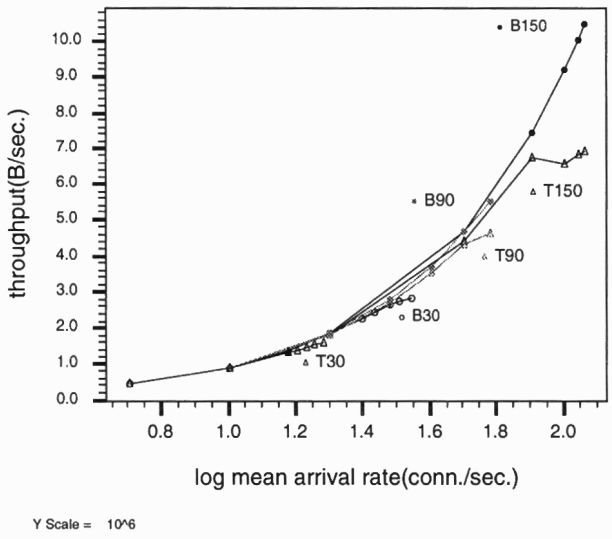


Figure 9: System throughputs in reference network

TCP tops out earlier than BLT. In other words, TCP arrives at the "knee" earlier. Figure 10 shows network power of TCP and BLT with different bottleneck sizes. In the graph, the initial alphabet represents the congestion control scheme used("T"=TCP, "B"=BLT), and the number after the initial means the bottleneck size in $Mbps$. The X axis represents λ in log scale, and the Y axis is

P also in log scale. As the figure clearly shows, our main thesis of this paper is that the power "generated" by conventional congestion control schemes compared to our approach will be

- less, and
- less scalable with bandwidth increase.

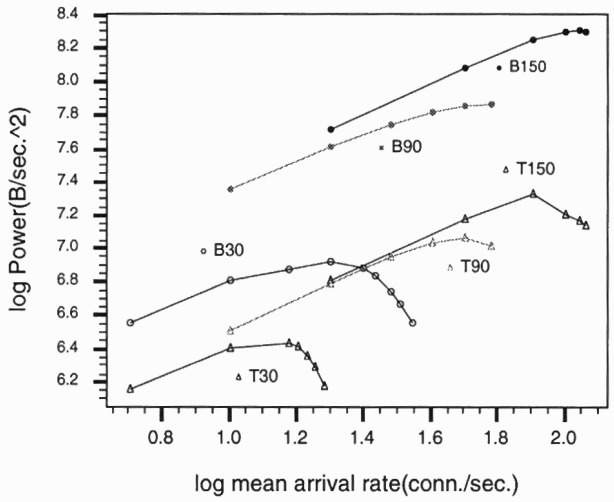


Figure 10: Network powers for various bottleneck sizes in the reference network

In Figure 10, the powers of TCP and BLT both increase with the increase of physical bottleneck bandwidth. But for all BW_b, the power of BLT is much higher, with the maximum ratio of roughly ten times of the TCP power at $BW_b = 150Mbps = BW_N$. Even BLT power at $BW_b = 90Mbps$ excels that of TCP at $BW_b = 150Mbps$. If we see the maximum powers, BLT increasingly outperforms TCP with the ratio of 3, 6, and 10(0.5, 0.8 and 1 in log scale) as BW_b increases. In the reference network with varying BW_b, BLT's performance and scalability are better than TCP's.

In terms of fairness, there is no clear winner. In Figure 11, the fairness is shown for BLT and TCP. For $BW_b = 30Mbps$ and $BW_b = 150Mbps$, BLT has better fairness, especially at higher load. But at $BW_b = 90Mbps$, TCP shows better result. This is surprising, because BLT switch discriminates against opportunistic packets while TCP does not assume anything on network switch for fairness. We suspect that the glitch in the switch queueing algorithm described in Section 3.3.2 is the cause of this phenomenon and it will be rectified in the future research. Overall, the best fairness is achieved when the load is the lowest.

In this section we varied BW_b, and in the next section we vary BW_N, F, and D from the reference network to see the performance scalability of each scheme.

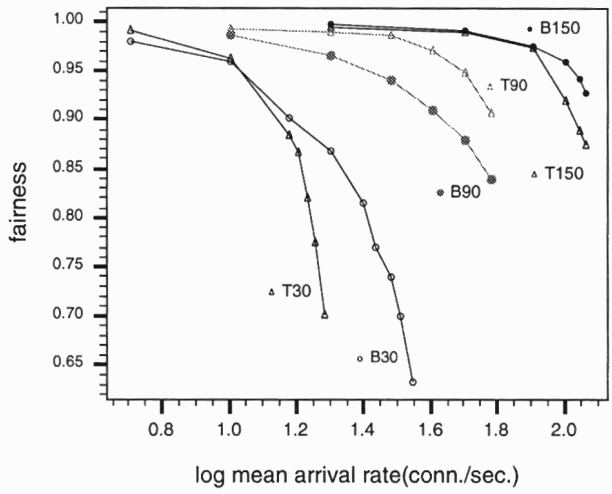


Figure 11: Fairness of TCP and BLT

4.2.2 Bandwidth scaling

In this section we vary BW_N to see the effect of larger network bandwidth on performance and scalability of the two schemes. Note that BW_b is automatically scaled with BW_N so that it ranges from $\frac{1}{5}$ to $\frac{3}{5}$ to 1 times of BW_N. We already confirmed that our conjecture on scalability in terms of BW_b; *i.e.*, with a given BW_N, TCP's P tops out much earlier and much less scalable than BLT's. Now we experiment with a larger network bandwidth of $BW_N = 1Gbps$.

Figure 12 shows results that are similar to those in Figure 10. For example, the power difference increases with larger BW_b. However, the important change is that the difference in power increase with the same $\frac{BW_b}{BW_N}$ between BLT and TCP has increased for all BW_b. For instance, at $BW_b = 200Mbps$, BLT maximum power has increased by 1.5 in log scale, but it is 1.1 in TCP. Similarly, it was 1.3 versus 0.8 and 1.1 versus 0.7 at $BW_b = 600Mbps$ and $BW_b = 1Gbps$, respectively. Table 1 shows that not only the increase of BW_b but also the increase of BW_N yields better power increase with BLT than TCP. Specifically, $\Delta logP_m(BLT) - \Delta logP_m(TCP) \geq 0.4$, meaning the power difference between BLT and TCP has gotten widened by another 2.5 to 3.5 times in network power with the bandwidth increase. This trend is strong evidence that in multi-gigabit networks, BLT performance will overshadow the performance of conventional dynamic control schemes. Figure 13 shows the fairness characteristics of the two scheme in $BW_N = 1Gbps$ environment. Only at $BW_b = 200Mbps$ TCP is comparable to BLT. As BW_b increases, BLT fairness is better than in TCP. This is close to our expectation. It seems that the fairness rapidly deteriorates around the arrival rate that achieves the maximum power. The maximum power is achieved when the system is on the "knee" and after that the fairness seems to suffer from excessive load.

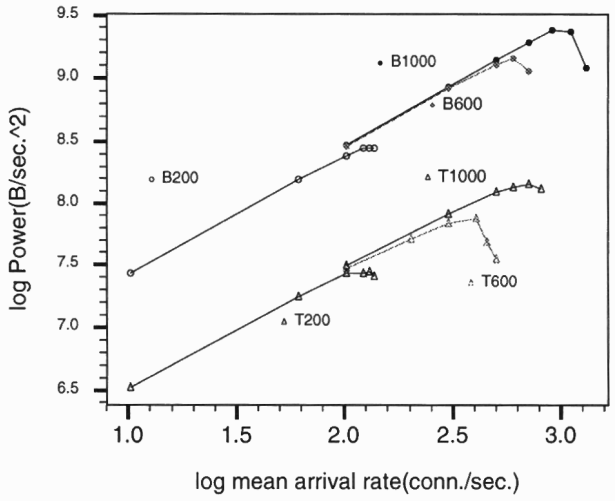


Figure 12: Network powers for various bottleneck size with $BW_N = 1Gbps$, $F = 100KB$

$\frac{BW_b}{BW_N}$	$logP_m(B/sec^2)$					
	BLT			TCP		
	$BW_N = 150Mbps$	$BW_N = 1Gbps$	$\Delta logP_m$	$BW_N = 150Mbps$	$BW_N = 1Gbps$	$\Delta logP_m$
$\frac{1}{5}$	6.9	8.4	1.5	6.4	7.5	1.1
$\frac{3}{5}$	7.9	9.2	1.3	7.1	7.9	0.8
1	8.3	9.4	1.1	7.4	8.1	0.7

Table 1: Maximum power(P_m) increase and scalability of TCP and BLT

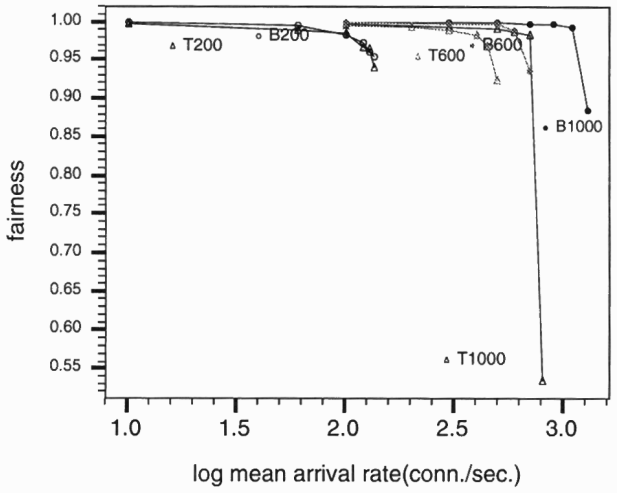


Figure 13: Fairness of TCP and BLT, $BW_N = 1Gbps$

Before proceeding further, here we make clear what we are pursuing throughout this series of experiments. In Figure 14, we put our experiments on the bandwidth-delay plane. We have seen that the Reference Network and Faster Network both yield far more power with BLT. So $(150Mbps \times 30ms \approx 0.6MB)$ and $(1Gbps \times 30ms \approx 4MB)$ are both beyond the boundary. We put them on the non-shaded part of the network, that represents the space where bandwidth-delay tradeoff approach is beneficial. The shaded area represents the bandwidth-delay space where the use of conventional dynamic control schemes is beneficial. The dotted lines connect equi-$BW \times D$ points. As we briefly mentioned in Section 3, our task is threefold. First, show the limitation of conventional approaches, second, propose a new scheme that overcomes the limitation, and third, determine the boundary on the bandwidth-delay plane between these two regions where one scheme outperforms the other. In Section 4.2.1 and 4.2.2 we show that BLT indeed excels and scales better than TCP in large bandwidth delay network. This addresses the first two tasks, so the remaining task is to find the boundary where BLT performs poorly when compared to TCP. It is evident that the boundary lies where the bandwidth-delay product is smaller than in the previous 2 experiments.

4.2.3 Scaling down network size

One way to reduce the bandwidth-delay product is to scale D down. A nationwide network size is $RTT \approx 30ms$ at its maximum, but smaller wide-area networks such as AURORA network will have $RTT \approx 3ms$ at the maximum[11]. In Figure 15, we show the network powers when $D = 3ms$. We

[11] However, $D = 3ms$ is almost twice as large as the average round-trip inter-nodal distance[JLK78] of AURORA network. The inter-nodal distance of AURORA network is around $1.55ms$.

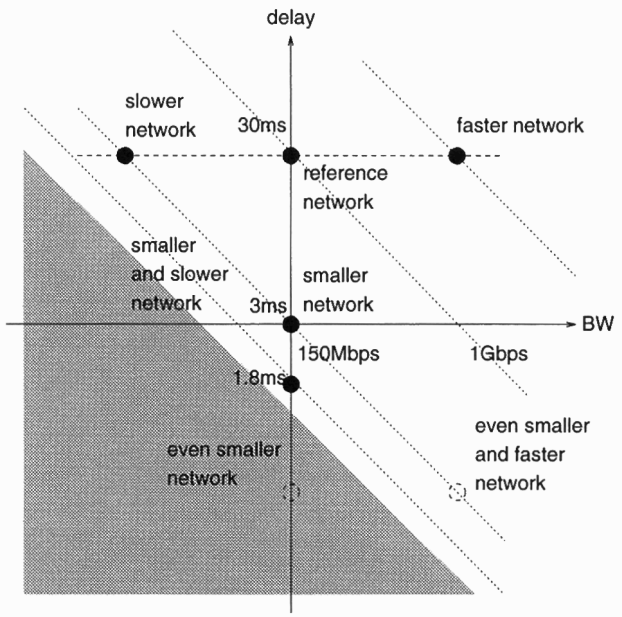


Figure 14: Bandwidth-delay space with $F = 100KB$

notice two major changes. First, the network power is overall larger than in the larger propagation delay network. This is because the response time and throughput both improve with smaller propagation latency. Therefore, the sender stalls shorter amount of time waiting for ACKs to arrive. When bandwidth-delay product is much higher than the window size, a data burst is only a blob on a long path[12]. After quickly transmitting a burst of packets, the sender stalls a long time, waiting for the group of ACKs arrive. Such inter-burst interval is shortened with small propagation latency, improving throughput and response time.

Second, the gap between the BLT power and TCP power narrowed. This shows that as bandwidth-delay product decreases, the relative performance gained through BLT gets smaller. But still, BLT's power is safely over TCP's, with increasing gap as BW_b increases. In Figure 16, we show the fairness of the two schemes. Overall, BLT fairness degrades with smaller bandwidth-delay product, if we ignore the indeterminate effect of the glitch of BLT switch queueing algorithm.

Since $D = 3ms$ did not push the BLT performance below that of TCP, we further reduce D and set $D = 1.8ms$. This amounts to about 170 *miles* in distance and the bandwidth-delay product of about $35KB$. As we expected, the result shown in Figure 17 tells us that generally the network power increased due to the shortened feedback loop. However, this is only visible when BW_b approaches BW_N. In addition, the gap between the powers slightly decreased in general. Figure 18 shows the fairness characteristics of BLT and TCP under this configuration. Again, we find it hard to generalize the result to any consistent tendency based on the bandwidth-delay product or

[12]Note that it does not have to do small receiver window. We already assumeed "infinite" receiver window(*e.g.* by adopting TCP Window Scale option[JBB92]) and only the congestion window size matters.

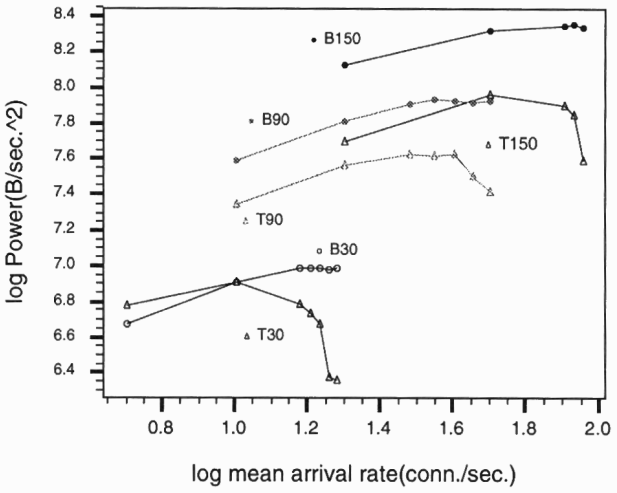


Figure 15: Network powers of TCP and BLT in a network with $D = 3ms$, $F = 100KB$ and $BW_N = 150Mbps$

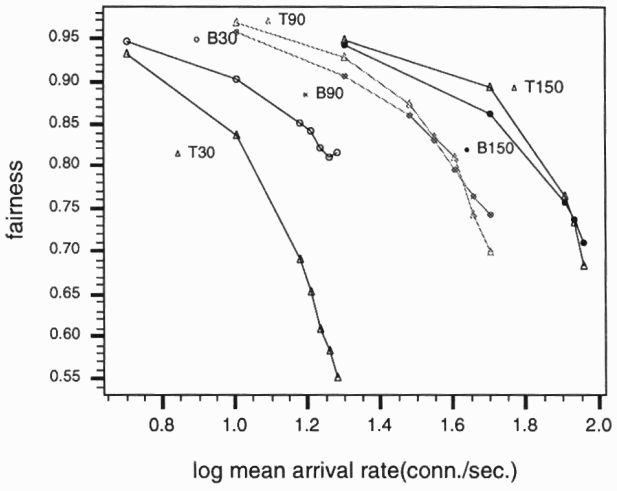


Figure 16: Fairness of TCP and BLT with the reference parameters, except $D = 3ms$

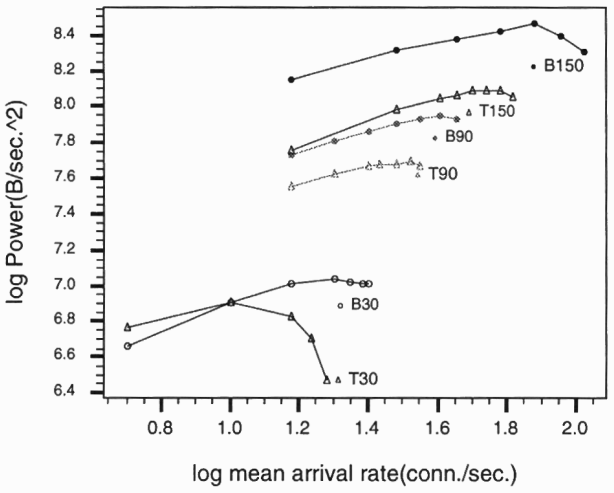


Figure 17: Network powers with in an AURORA-size network with $BW_N = 150Mbps$, $F = 100KB$

on some other parameter.

In the current configurations, even smaller network size implies impractical small buffer size at the bottleneck switch, and we defer the experiment on such cases to future work where we will change the buffer size configuration. If we reduce D to $1ms$, then the buffer size in terms of packets in $S1$ at BW_d will be 2 or 3. Under such condition, comparing TCP and BLT is almost meaningless, as well as impractical. For comparison, ATM Forum's rate-based flow control scheme[YH94] assumes multiple (end-to-end round-trip time $\times$ bandwidth) worth of buffer space at each switch. In future investigation, we will assume larger switch buffer size.

4.2.4 Slower network

Another way to reduce bandwidth-delay product is to decrease BW_N. Hopefully, the boundary between BLT-beneficial area and the other area will be drawn based only on the bandwidth-delay product, and we expect to get the same result when we reduce BW_N to yield the same bandwidth-delay product as in the previous experiment. To compare with ($D = 3ms$, $BW_N = 150Mbps$) case, we set $BW_N = 15Mbps$ and $D = 30ms$. Figure 19 shows the power characteristics, and we see that the power is lower about 2 orders of magnitude smaller compared with AURORA-sized network case. This is roughly explained by the fact that the raw bandwidth was decimated, while the feedback loop length was increased tenfold. We also see that when $BW_b = 3Mbps$ TCP power is slightly better with low arrival rate. But this soon vanishes with increased arrival rate and/or larger BW_b. We can safely conclude that this result supports the benefits of BLT as we had hoped. Figure 20 shows the fairness characteristics.

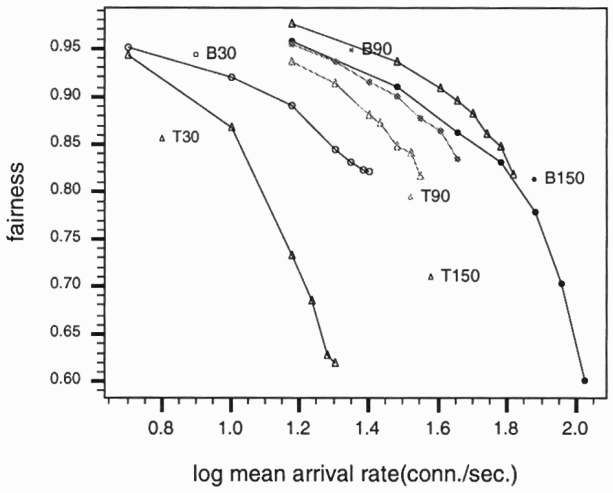


Figure 18: Fairness of TCP and BLT, $D = 1.8ms$

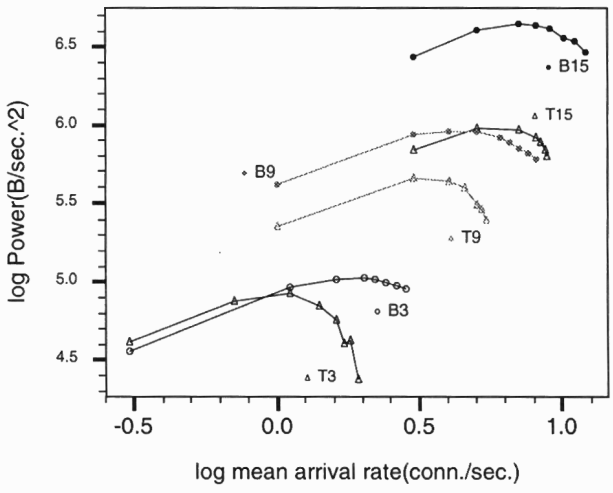


Figure 19: Powers of TCP and BLT, $BW_N = 15Mbps$

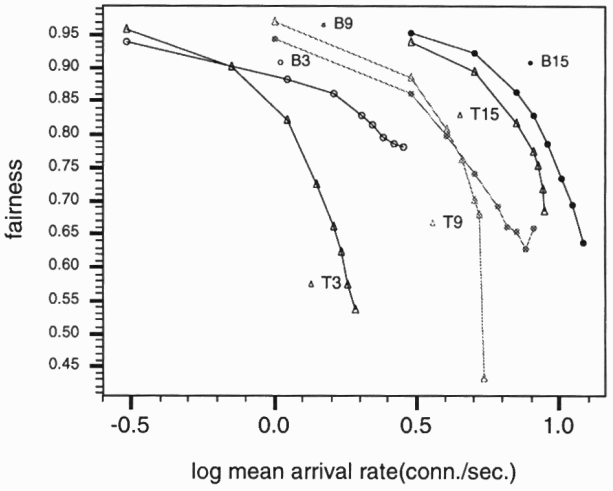


Figure 20: Fairness of TCP and BLT, $BW_N = 15Mbps$

4.2.5 Scaling data size

So far we have seen the performance with $F = 100KB$ only. In this section, we vary F for all three previous combinations of bandwidth-delay parameters. As far as the bandwidth-delay plane is concerned, we are now looking at a different plane, and the boundary between the two areas could shift. Other parameters we previously fixed, such as switch buffer size for one, can also significantly affect the boundary. In short, every one of the network configuration parameters, except bandwidth and delay, will create a new plane where the boundary is shifted.

We expect as file size grows, the performance difference between BLT and TCP will narrow down because the relative overhead of dynamic window sizing such as Slow Start will be smaller in long transmissions. Figure 21 shows the power characteristics of TCP and BLT with $F = 1MB$. Overall power decreases compared to the reference network. In case of BLT, the maximum power in all bottleneck size fell at least one order of magnitude. In TCP, the decrease is milder, about half order of magnitude. The nominal cause of the overall drop is the increase of response time R. With F ten times larger than in the reference network, we can expect that the response time will roughly increase tenfold. This amounts to one order decrease in power, and that is precisely what happened to BLT. The fact that the increase of F has direct impact on BLT response time(and power) means there are little overhead factors in BLT that would melt away with the increase in file size. However, for TCP, as the overhead of dynamic window size maneuvering decreases, the drop was milder. This again clearly shows how more efficient BLT is than TCP.

In general, the usefulness of BLT decreases when file size increases, especially when λ is very small making the probability of congestion low. In such cases TCP can even outperform BLT

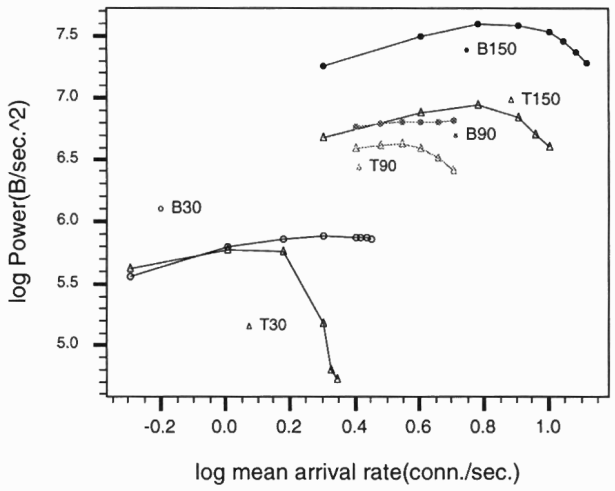


Figure 21: Network powers in transfer of 1MB file in reference network

when $BW_b = 30Mbps$. This is because packet drops due to congestion is rare at low arrival rate, improving TCP throughput. However when BW_b increases, the bandwidth-delay product increases and even with very low arrival rate, TCP cannot outperform BLT. And with the increase of λ, even when the bandwidth-delay product small, TCP rapidly deteriorates. Figure 22 shows fairness values in this environment. As the increase of BW_b yields better performance for BLT, so does the increase of BW_N. When we increase BW_N to $1Gbps$, we see that the power difference between BLT and TCP widens again(Figure 23). This is the strong point of BLT. It scales better with bandwidth. Figure 24 shows fairness characteristics. The disadvantage of conventional schemes like TCP is amplified in small file transfers. Almost all data transfer protocols adopt some form of Slow Start, and adaptive rate searching algorithm, and this search overhead is critical to small files because for them the time spent in adjustment could be very long. Here, we decreased F to $10KB$ in the reference network. The result shown in Figure 25 seemingly disagrees with our expectation that the power will increase tenfold, but this is because we calculated the response time to be $T_f - T_i$ where T_f is the time that the acknowledgement for the last bit of the sent file arrives at the sender. In the current setting, the entire file transmission time in BLT is much smaller than the time that the source waits for the last ACK. Namely, the file transmission time for 20 $packets(10KB)$ is less than $1ms$ where the ACK travel time is $15ms$. This is not the case for TCP, where it goes through Slow Start and other rate adjustment at least for 4 RTT. If we take this into account, we get the result as shown in Figure 26. Now we can clearly see that

$$\Delta log P_m(BLT) \geq \Delta log P_m(TCP)$$

which means BLT is more scalable. Because file transfer time is much smaller than the propagation

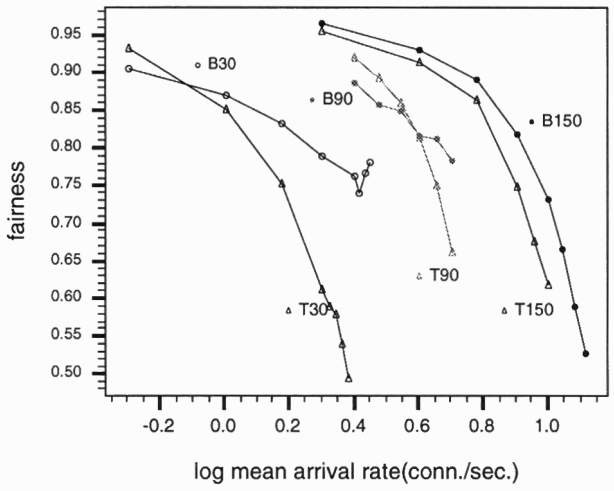


Figure 22: Fairness of TCP and BLT, $F = 1MB$

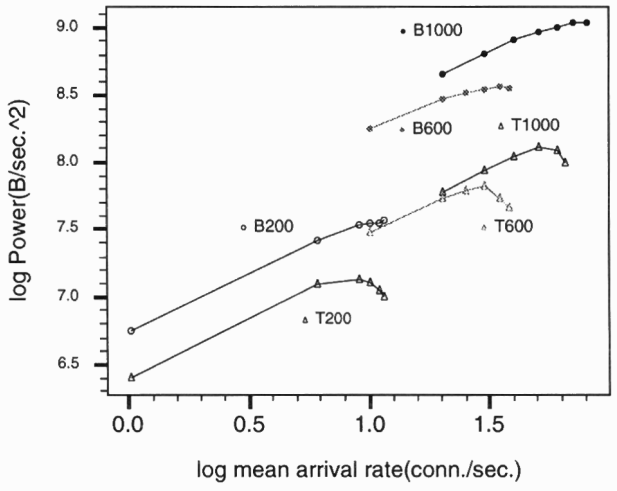


Figure 23: Network powers in transfer of $1MB$ file in $1Gbps$ network

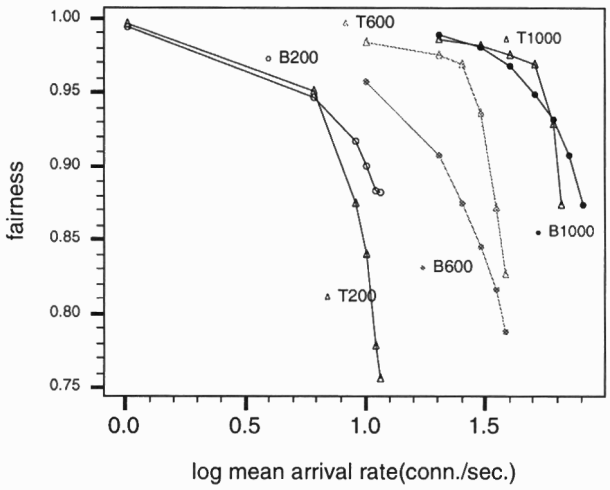


Figure 24: Fairness of TCP and BLT, $BW_N = 1Gbps,\ F = 1MB$

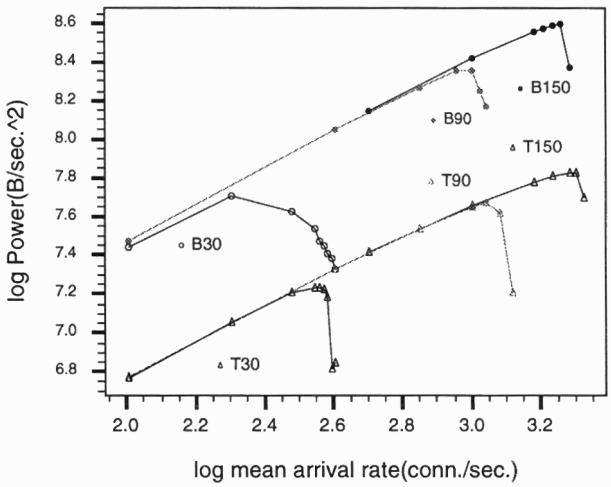


Figure 25: Network powers with reference parameters except $F = 10KB$

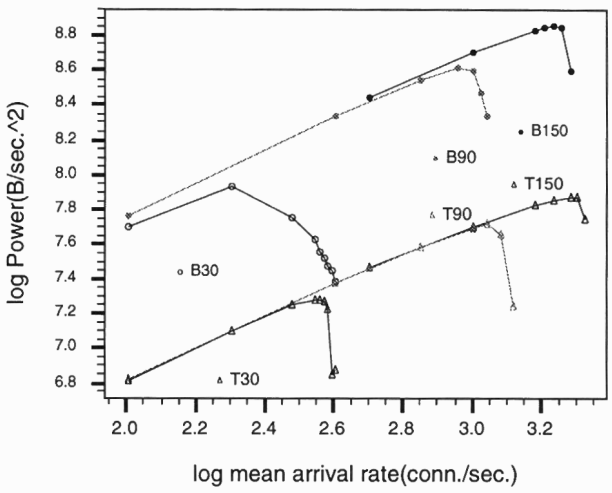


Figure 26: Network powers considering ACK overhead

delay, the file size decrease to one tenth does not result in 10 times power increase. For instance, if only one RTT is added to R because of a packet drop, it will drastically pull the power down. Fairness results are shown in Figure 27.

4.2.6 Comparison with no congestion control

For completeness, we compare the performance of BLT and TCP with that of what we call "Bully-Tyrant" scheme that does not exercise any flow control. Bully-Tyrant(BT) behaves similar to BLT except that it does not provide any information on each packet's degree of opportunism. In some sense, it is also similar to old TCPs where no congestion control algorithm was used. We intend to show that aggressive exploitation of bandwidth alone cannot achieve comparable performance as BLT. In Figure 28 we can see that aggressive exploitation of bandwidth pays off, and BT power almost reaches BLT power, well exceeding TCP. There is little difference between BLT and BT in powers when λ is relatively small. However, as the arrival rate increases, "congestion meltdown" occurs earlier in BT while BLT gracefully adapts to load increase. Figure 28 is interesting in that it shows how much aggressive bandwidth exploitation boosts performance. Conversely, it shows the failure of conservativeness of dynamic feedback control in high-speed networks. But in terms of fairness, we can easily expect that BT will mark the lowest and it does in Figure 29.

4.3 Summary

In a simplified and generalized network environment we measured the (maximum) network powers of TCP and BLT. It is clear that BLT yields far more power than TCP and it scales much better

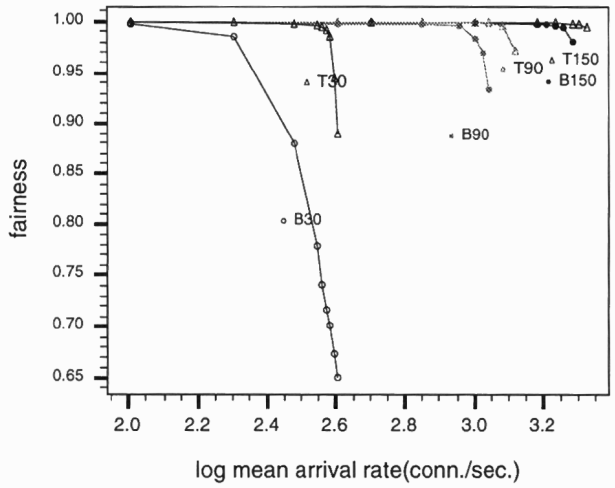


Figure 27: Fairness of TCP and BLT, $F = 10KB$

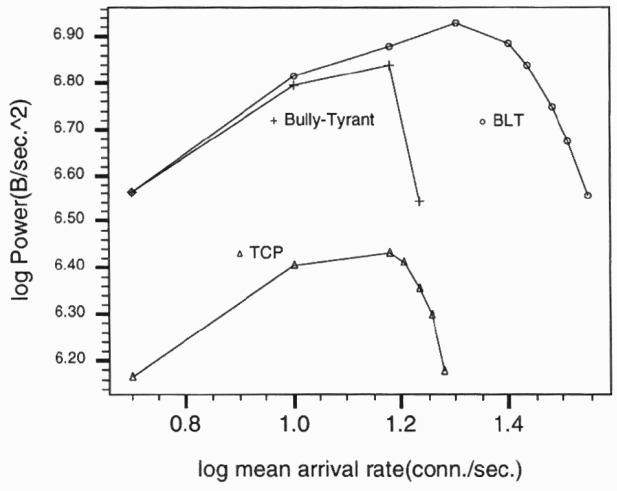


Figure 28: Network power of TCP, Bully-Tyrant and BLT for $BW_b = 30Mbps$

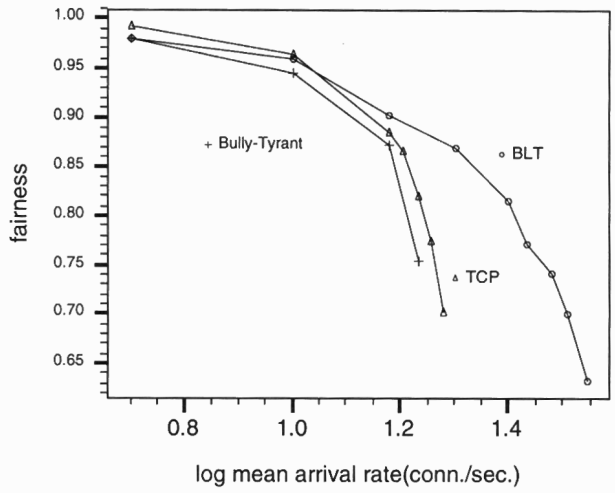


Figure 29: Fairness of TCP, Bully-Tyrant and BLT for $BW_b = 30Mbps$

than TCP with bandwidth. In terms of fairness, however, BLT and TCP showed mixed results. We suspect BLT's fairness improve by correcting the switch queueing discipline, but we have yet to see how much improvement can be achieved. In one of our experiments, we saw that the main driving force behind BLT's good performance is the aggressiveness in utilizing bandwidth. We think this is a very important lesson in designing congestion control schemes for high-speed networks in the future.

4.3.1 Powers at a glance

In Figure 30, we summarize the maximum powers for all parameter combinations we experimented for $F = 100KB$. The powers are in log scale, so the bottom parts not shown in the graph are insignificant, and only the difference between the powers of BLT and TCP for each network configuration matters. We see that the difference in power between TCP and BLT widens, with both BW_b and BW_N. In small network with less D, BLT still outperforms TCP but the power difference reduces. But even in that case, as BW_b increases, the difference grows again. Because P is in log scale we are likely to be deceived on the real difference between the performances. In Figure 31, we show only the case of $BW_N = 1Gbps$ with normal scale. And now we can see the difference in performance and scalability. The main factor that resulted in the huge difference, seems to be R rather than Θ. For instance, when $BW_b = 1Gbps$ $\Theta_B \approx 605Mbps$ while $\Theta_T = 318Mbps$, which is about two times difference, but $R_B = 0.032s$ and $R_T = 0.2894s$, about nine times difference. So the difference in power is about 17 times, which appeared as about 1.23 order of magnitude in log scale in Figure 30. Note that the Θ's are *system* throughputs, and they are different from

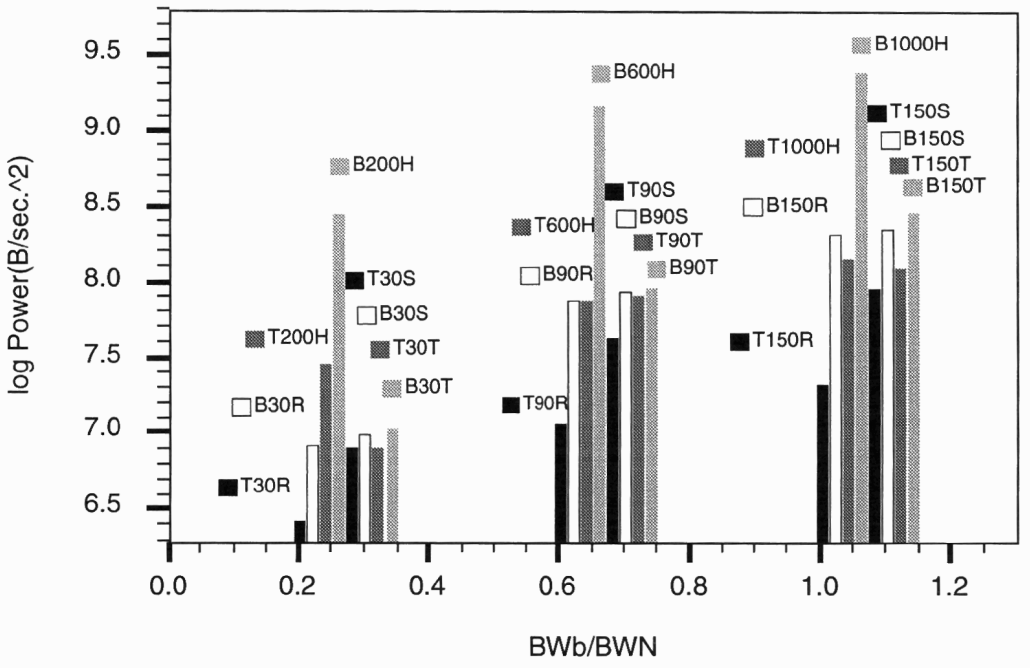


Figure 30: Maximum powers in log scale, $F = 100KB$

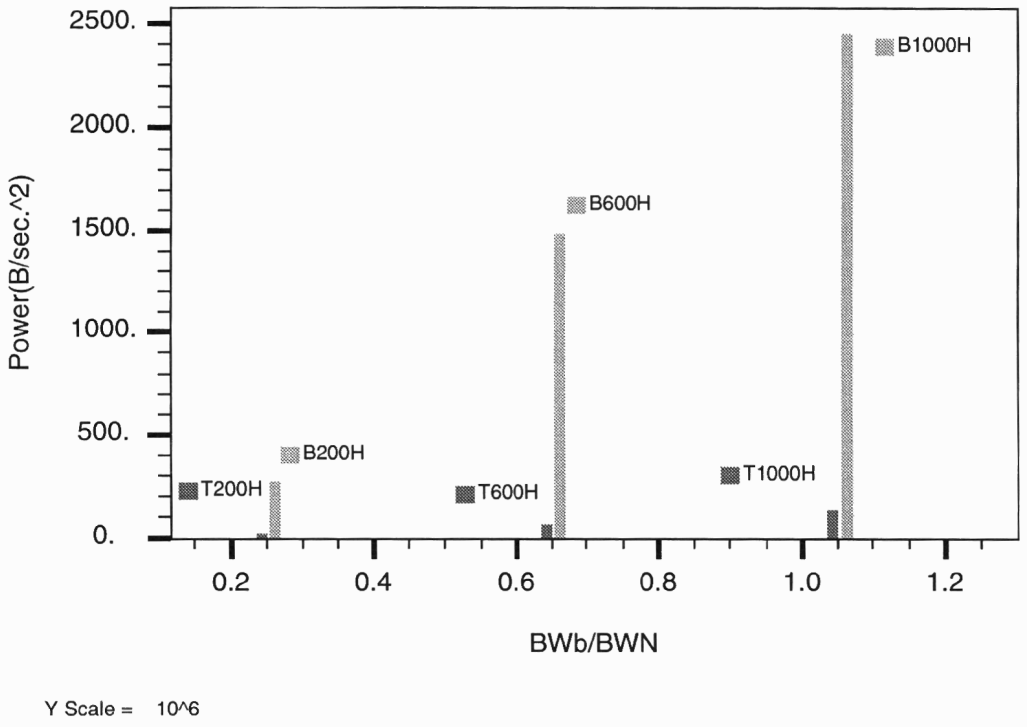


Figure 31: Maximum powers in normal scale, $F = 100KB$, $BW_N = 1Gbps$

the throughputs viewed from individual connections. Since TCP connections linger longer in the system, the system carries more connections than in BLT, and the throughputs seen by individual connections are much smaller. But there are many outstanding connections, and the aggregate throughput(*i.e.*, system throughput) is is only half of BLT's. In the above example, the average individual throughput of TCP case would be

$$\begin{aligned} \theta_T &= \frac{F}{R_T} \\ &= \frac{100KB}{0.2894s} \\ &\approx 2.8Mbps \end{aligned}$$

This implies that there were 115 outstanding connections on the average in the TCP case. On the other hand, in BLT case there were 29 outstanding connections on the average. This means that $\theta_B = 20.8Mbps$, and $R_B = 0.0385s$ which roughly coincides with the measured number above[13]. There was no such error in measuring R, so if we use $R_B = 0.032$ we get $\Theta_B = 725Mbps$, and this nicely matches with $\lambda = 900\ conn./s = 720Mbps$ where the highest power was achieved, and the system is stabilized.

4.3.2 Drawing boundary with bandwidth-delay product

We also attempted to show that we can determine the boundary between the two areas where BLT or TCP outperform each other. With current network configuration where the network switches are allowed to have one (bandwidth $\times$ link delay) worth of buffer space and packet size is $500B$, it was impractical to experiment on very small bandwidth-delay product networks. But down to $35KB$, BLT outperformed TCP. It is still possible that BLT can outperform TCP in smaller bandwidth-delay product networks than this, but we leave it to future work.

4.3.3 Remaining works

In future experiments, we will modify the BLT switch algorithm so that when the queue is full and new packet is coming in, the lowest priority packet be dropped. For TCP, we will fully implement Vegas TCP which is said to improve throughput by 40 to 70% over Reno TCP(although the experiment was done in a relatively low-speed network).

In this experiment we intentionally used the simplest network model, because we wanted to observe the affects of bandwidth-delay product without muddled by other small parameters. For instance, there was only one bottleneck. We will add more bottlenecks along the path of connections in future works.

One of the parameters we fixed throughout the experiments was switch buffer size. However, this parameter can significantly affect performance because it changes the network characteristics

[13]This discrepancy stems from the error in calculating Θ. We used the longest run time among three independent runs in calculating Θ, and the above Θ's are from the pessimistic case(this will be rectified in future simulations).

from the viewpoint of the congestion control. In most cases, a buffer size of one (bandwidth $\times$ link delay) is too small in reality. For instance, rate-based ATM flow control deploys multiple (bandwidth $\times$ round trip delay) worth of buffer at each switch node. In future works, we will also investigate the scalability of the congestion control schemes with buffer size.

As we briefly mentioned in the simulation assumptions, we limited the traffic characteristics to file transfer. In the future, we will incorporate other traffics such as intermittent and interactive traffics. It is also of interest to have self-similar cross traffic at the switches, to see how the congestion control schemes react to such changes at the switches.

5 Conclusion

5.1 Summary

In this paper, we addressed the issue of scalability in controlling congestion for high-speed networks. To the best of our knowledge, there has not been any serious efforts to determine how existing congestion control schemes will scale with increasing bandwidth in high-speed networks. The existing schemes use some form of dynamic rate adaptation mechanisms such as Slow Start and/or additive-increase/multiplicative-decrease, whether it is a window-based scheme or a rate-based scheme[RJ90, Jac88, MK92, YH94]. These dynamic rate control schemes are bound to delay in that the efficiency of control is determined by the fast feedback of network load. But the length of feedback loop logically increases with bandwidth increase, decreasing their efficiency. To overcome the problem of increasing feedback delay, we need an entirely different way of thinking. We propose an approach based on bandwidth-delay tradeoff, called BLT. While conventional schemes do the control in time domain, we propose to do the control in bandwidth domain. In other words, conventional schemes change window size over time based on the most recent feedback information from the network. In BLT, window sizes are changed in bandwidth domain without the help of feedback information. Conceptually, since the change of window size occurs in the bandwidth domain, the window change occurs at the same instance in time. So the traffic source sends out the whole spectrum of windows at the same time. It takes more bandwidth to create this spectrum of windows, but it can help alleviate the delay problem. The bottleneck switch chooses the right window size among the array of window sizes, discarding all others. Since the switch does not need communication with the traffic source to negotiate on the window size, there is no problem of delay. Moreover, since BLT is designed to scale *with* bandwidth, it does not have the scalability problem as conventional control schemes. The source of longer feedback loop problem was the bandwidth increase, but BLT is only better with the increase.

We had two goals when we set out this investigation. First, we intended to show the scalability problem with existing schemes. Second, we intended to show the feasibility of our new approach. Third, we wanted to draw a line in bandwidth-delay space, if it exists, which marks the point from which BLT performs better than conventional schemes. The first and the second goal were achieved

by comparing the performance and scalability of TCP and the window-based implementation of BLT. In all experimentable ranges, from small bandwidth-delay product to large bandwidth-delay product, BLT outperformed TCP approximately around 10 times. More importantly, BLT performance increases faster than TCP with bandwidth increase. We have experimented up to $1Gbps$ network, but it is of interest whether or not this improvement in scalability of BLT can be sustained. We expect that it can, and will further outperform TCP. With the third goal, we faced some practical problems in the experiment. But down to $35KB$ of bandwidth-delay product, BLT outperformed TCP. From the last experiment presented in Section 4.2.6, we learned that *aggressiveness* is the key to harvest good performance in future high-speed networks. We can characterize conventional approaches to traffic control as the marriage between "fragile network" and "cautious user". But we believe this should be changed to "abusive user" that is aggressive to use idle bandwidth and "rugged network" that can readily control excessive aggressiveness. One major issue would be how to guarantee fairness in this new environment.

This study explored the feasibility of the bandwidth-delay tradeoff approach, and in the process we made many assumptions and waived many practical considerations. Thus, quite a few issues should be resolved for BLT to be deployed in real networks, but it is clear that bandwidth-delay tradeoff is a new promising approach to traffic control in future high-speed networks since the feasibility and the potential of this approach in terms of performance and scalability showed promise.

An added advantage is that the improvements are not limited to a particular layer or to a particular protocol. Furthermore, not only window control schemes but also rate based schemes and ATM flow control schemes also adopt dynamic control in one way or another, and the implication of this experiment also applies to them.

5.2 Future works

5.2.1 Rate-based BLT

BLT is not inherently limited to window-based implementation. Window-based control has a problem of "lock-step" transmission, where the sender transmits a burst and then waits for the group of ACKs for a long time and transmits again. Due to this reason, we expect that rate-based traffic control schemes will eventually replace window-based schemes. Because BLT is a methodology to solve the scalability problem of dynamic flow/congestion control, there is no reason why it cannot be applied to dynamic rate-based traffic control schemes. Recently ATM Forum voted to support one of the rate-based flow control proposals[YH94, New93] and it is expected to become the major flow control mechanism for ATM networks. One notable thing about the scheme is that it uses additive-increase/multiplicative-decrease for rate adjustment at the traffic source, with the feedback provided by forward/backward explicit congestion notification. Yin and Hluchjy[YH94] also alludes to some use of Slow Start, although the detail has yet to be worked out. In essence, there is little conceptual difference between window-based control and rate-based control. We expect the scheme to face the same control delay problem when it is used for large bandwidth-delay product network,

and this issue is already under investigation. Although the basic philosophy behind window control and rate control is similar, we see an array of difficult problems. For example, to apply BLT to rate-based control, one should be able to create multiple rates in a stream just as multiple windows with different sizes were created in window-based BLT, which is tricky. Even if we can create this multiple-rate stream, sequence number synchronization among these substreams and between the sender and the receiver is a difficult problem.

5.2.2 Self-similar interference

We have already pointed out that self-similarity of data traffic can be a major problem for conventional traffic control schemes that are becoming ever slower due to increasing bandwidth. One of the implications of self-similarity is that the burstiness will still appear in small time scales well under round-trip delays of wide-area networks. We expect that the conventional schemes will fail to react to, and exploit, the fluctuation in the network load. On the other hand, BLT carries more than sufficient amount of data up to the bottleneck, which enables a BLT stream to quickly exploit dynamically available bandwidth. Specifically, BLT makes the switch to take in the maximum amount of data from its data stream when the stream passes. Conversely, when the network load quickly surges the network switch can conveniently discard opportunistic packets to control the congestion and distribute its bandwidth fairly among the connections. In our continuing investigation, we will employ a self-similar cross traffic at bottleneck switches and see how conventional schemes and BLT react to such network condition.

5.2.3 Other switching disciplines

In this paper, we assumed a hybrid of FIFO and priority queueing at network switches. In principle, BLT does not limit the switch queueing discipline to one or two, provided that some form of discriminating action is exercised based on the degree of opportunism of packets. We will investigate if other queueing disciplines can be coupled with BLT sources in future work.

5.2.4 Bandwidth usage optimization

An obvious optimization target in BLT is in its bandwidth usage. Since this investigation is a feasibility study we do not limit the bandwidth use in any way, but in reality this is somewhat unrealistic even if we are given a large bandwidth. Although we still believe that the bandwidth requirement of BLT would not be that excessive because it only uses the bandwidth that would be idled with conventional control schemes, we will investigate ways to further decrease the requirement.

5.2.5 Deploying in real world

One last issue we have to deal with is how BLT can interface with the real networks, especially with underlying ATM-based network fabric. Separate from our investigation on rate-based BLT in ATM

networks, we will have to solve many possible problems. For instance, if BLT is used on transport layer[14], priority discarding should also be exercised on lower layers because otherwise multiplexing points on lower layers will be flooded with opportunistic BLT packets. This will block not only other connections but we found that BLT itself cannot achieve satisfactory performance, either. And again, the underlying networks won't eagerly drop packets, as we can see in ATM rate-based flow control. We will have to deal with many such problems to transform BLT from a theoretical solution to a practical traffic control mechanism.

References

[BD94] C.E. Bagwell and J.N. Daigle. ATM Cell Delay and Loss for Best-Effort TCP in the Presence of Isochronous Traffic. In *Proceedings of IEEE INFOCOM*, 1994.

[BOP94] L.S. Brakmo, S.W. O'Malley, and L.L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proceedings of ACM SIGCOMM*, 1994.

[CDJM91] Ramon Caceres, Peter B. Danzig, Sugih Jamin, and Danny J. Mitzel. Characteristics of Wide-Area TCP/IP Conversations. In *Proceedings of ACM SIGCOMM*, pages 101–112, 1991.

[Cha91] J. Chao. A Novel Architecture for Queue Management in the ATM Network. *IEEE Jounal on Selected Areas in Communications*, 9(7):1110–1118, September 1991.

[CLZ87] D. D. Clark, M. L. Lambert, and L. Zhang. NETBLT: A bulk data transfer protocol. In *Proceedings of the ACM SIGCOMM symposium*, 1987.

[Cro94] J. Crowcroft. Why Lossy Internetworking and Lossless ABR ATM Services Do Not Go Together, April 1994. Comments on ATM flow control mechanisms, comp.dcom.cell-relay newsgroup.

[Fel93] D.C. Feldmeier. A Framework of Architectural Concepts for High-Speed Communication Systems. *IEEE Jounal on Selected Areas in Communications*, 11(4):480–488, May 1993.

[FJ93] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[Flo94] S. Floyd. Tcp and explicit congestion notification, 1994. Submitted to ACM CCR.

[FRW92] K. Fendick, M. Rodrigues, and A. Weiss. Analysis of a Rate-Based Control Strategy with Delayed Feedback. In *Proceedings of ACM SIGCOMM*, pages 136–148, 1992.

[14] Note that BLT can be used on any layer that needs traffic control.

[GHKP78] A. Giessler, J. Hanle, A Konig, and E. Pade. Free Buffer Allocation – An Investigation by Simulation. *Computer Networks*, 1(3):191–204, July 1978.

[GW94] M.W. Garrett and W. Willinger. Analysis, Modeling and Generation of Self-Similar VBR Video Traffic. In *Proceedings of ACM SIGCOMM*, 1994.

[HKM93] E.L. Hanhe, C.R. Kalmanek, and S.P. Morgan. Dynamic window flow control on a high-speed wide-area data network. *Computer Networks and ISDN Systems*, 26:29–41, 1993.

[Jac88] V. Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM*, August 1988.

[Jac90] V. Jacobson. Author's original explanation of Fast Recovery algorithm, 1990. end-to-end mailing list(end-to-end@ISI.edu).

[Jai90] Raj Jain. Congestion Control in Computer Networks: Issues and Trends. *IEEE Network Magazine*, pages 24–30, May 1990.

[Jai91] R. Jain. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, Inc., 1991.

[JBB92] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323, Network Information Center, SRI International, May 1992.

[JLK78] D.S. Johnson, J.K. Lenstra, and A.H.G. Rinnooy Kan. The Complexity of the Network Design Problem. *Networks*, 8:279–285, 1978.

[JR88] R. Jain and K.K. Ramakrishnan. Congestion Avoidance in Computer Networks with a Connectionless Network Layer: Concepts, Goals and Methodology. In *Proceedings of the Computer Networking Symposium*, pages 134–143, 1988.

[KBC94] H. Kung, T. Blackwell, and A. Chapman. Adaptive credit allocation for flow-controlled vcs. Technical report, ATM Forum, March 1994. available via anonymous FTP from virtual.harvard.edu:pub/htk/atm-forum/atm_forum_94-0282.ps.

[KC94] H. Kung and A. Chapman. The fcvc(flow-controlled virtual channels) proposal for atm networks. Technical report, ATM Forum, January 1994. available via anonymous FTP from virtual.harvard.edu:pub/htk/atm-forum/fcvc.ps.

[KKM92] H. Kanakia, S. Keshav, and P. Mishra. A Benchmark Suite for Comparing Congestion Control Schemes, 1992. AT&T Bell Labs., Work Project No. 311407-7307.

[Kle78] L. Kleinrock. On Flow Control in Computer Networks. In *Proceedings of ICC*, volume 2, pages 27.2.1–27.2.5, 1978.

[Kle79] L. Kleinrock. Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications. In *Proceedings of ICC*, pages 43.1.2–43.1.10, 1979.

[Kle92] L. Kleinrock. The Latency/Bandwidth Tradeoff in Gigabit Networks. *IEEE Communications Magazine*, pages 36–40, April 1992.

[KMT93] K.T. Ko, P.P. Mishra, and S.K. Tripathi. Interaction among virtual circuits using predictive congestion control. *Computer Networks and ISDN Systems*, 25:681–699, 1993.

[LTWW93] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the Self-Similar Nature of Ethernet Traffic. In *Proceedings of ACM SIGCOMM*, pages 183–193, 1993.

[MK92] P. Mishra and H. Kanakia. A Hop by Hop Rate-based Congestion Control Scheme. In *Proceedings of ACM SIGCOMM*, pages 112–123, 1992.

[New93] P. Newman. Backward explicit congestion notification for ATM local area networks. In *Proceedings of IEEE GLOBECOM*, pages 719–723, 1993.

[New94] P. Newman. Traffic Management for ATM Local Area Networks. *IEEE Communications Magazine*, pages 44–50, August 1994.

[oAml94] IP over ATM mailing list. Discussions on tcp/ip performance on atm networks, 1994. ip-atm@hplms2.hpl.hp.com.

[Pax94] V. Paxson. Growth Trends in Wide-Area TCP Connections. *IEEE Network Magazine*, pages 8–17, July/August 1994.

[PF90] D.W. Petr and V.S. Frost. Optimal Packet Discarding: An ATM-Oriented Analysis Model and Initial Results. In *Proceedings of IEEE INFOCOM*, 1990.

[RF94] Allyn Romanow and Sally Floyd. Dynamics of TCP Traffic over ATM Networks. In *Proceedings of ACM SIGCOMM*, August 1994.

[RJ90] K. Ramakrishnan and R. Jain. A binary feedbak scheme for congestion avoidance in computer networks. *ATM Transactions on Computer Systems*, 8(2):158–181, 1990.

[SCL93] S. Shenker, D.D. Clark, and L.Zhang. A Service Model for an Integrated Services Internet, October 1993. draft-shenker-realtime-model-00.txt.

[Ste94] W. Richard Stevens. *TCP/IP illustrated.* Addison-Wesley Pub. Co., Reading, Mass., 1994.

[SZC91] S. Shenker, L. Zhang, and D.D. Clark. Some Observations on the Dynamics of a Congestion Control Algorithm. In *Proceedings of ACM SIGCOMM*, August 1991.

[THP93] L. Tassiulas, Y. Hung, and S.S. Panwar. Optimal buffer control during congestion in an ATM network node. In *Proceedings of IEEE GLOBECOM*, 1993.

[Tou93] J.D. Touch. Parallel Communication. In *Proceedings of IEEE INFOCOM*, 1993.

[YH94] N. Yin and M. Hluchyj. On closed-loop rate control for atm cell relay networks. In *Proceedings of IEEE INFOCOM*, 1994.