Two-Way TCP Traffic over Rate Controlled Channels: Effects and Analysis

Lampros Kalampoukas*, Anujan Varma*
and
K. K. Ramakrishnan†

UCSC-CRL-96-23
August 21, 1997

* Board of Studies in Computer Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064

†AT&T Labs-Research
Florham Park, NJ 07932

This research is supported by the Advanced Research Projects Agency (ARPA) under Contract No. F19628-96-C-0038 and by the NSF Young Investigator Award No. MIP-9257103. We thank MIL3 for donating to us the OPNET modeling and simulation tool.

Abstract

We examine the performance of bidirectional TCP/IP connections over a network which uses rate-based flow and congestion control mechanisms. An example of such a network is an Asynchronous Transfer Mode (ATM) network using the Available Bit Rate (ABR) service. The sharing of a common buffer by TCP packets and acknowledgements has been known to result in an effect called *ack compression*, where acks of a connection arrive at the source bunched together, resulting in unfairness and degraded throughput. It has been the expectation that maintaining a smooth flow of data using rate-based flow control would mitigate, if not eliminate, the various forms of burstiness experienced with the TCP window flow control. However, we show that the problem of TCP ack compression re-appears even while operating over a rate-controlled channel, although the queues are primarily at the end systems now. By analyzing the periodic bursty behavior of the source IP queue, we are able to predict the peak values for the queue and arrive at a simple robust predictor for the degraded throughput, applicable for relatively general situations. The degradation in throughput due to bidirectional traffic can be significant. For example, even in the simple case of symmetrical connections with adequate window sizes, the throughput of each connection is only 66.67% of that under one-way traffic.

We validate our analysis using simulation on an ATM network using the Explicit Rate option of the ABR service. We show that the analysis predicts the behavior of the queue and the throughput degradation not only in simple configurations, but also in more general situations, such as multiple synchronized connections between a pair of end-systems, multiple end-systems having different round-trip delays, and in configurations with significant amounts of cross-traffic. We observe the need to separate the flow of acknowledgments and data for the bidirectional TCP connection and for interleaving their processing at the end-systems to overcome the problem of ack compression. We show that the rate-controlled channel localizes the effect of TCP burstiness and the accompanying dynamics to the outgoing queue at the source, requiring source-based approaches to reduce ack compression.

Keywords: rate control, TCP over ATM, congestion control, two-way traffic

1 Introduction

The Transmission Control Protocol (TCP) has become the most widely used transport-layer protocol today, due largely to the explosive growth of the TCP/IP Internet in recent years. An important component of TCP is the collection of algorithms used to perform congestion control and recovery [1, 2]. These algorithms give rise to a variety of interesting dynamics, some of which have been studied extensively [3, 4, 5, 6]. In this paper, our interest is in analyzing the dynamics of TCP connections in an Asynchronous Transfer Mode (ATM) network in the presence of two-way traffic.

We define *two-way* or *bidirectional* traffic as the traffic pattern resulting from two or more TCP connections transferring data in opposite directions between the same pair of end nodes over a network path. The TCP segments transmitted by the connections in one direction share the same physical path with the acknowledgements (acks) of connections in the opposite direction. These packets and acknowledgements may share a common buffer in the end systems as well as network switches/routers. This sharing has been shown to result in an effect called *ack compression*, where acks of a connection arrive at the source bunched together [6, 7]. The result of ack-compression is a marked unfairness in the throughput received with competing connections, and reduced overall throughput compared to what could be expected without this effect [7]. Ack compression may occur either at the end system or in a switch/router. In either case, the smooth flow of acknowledgements to the source is disturbed, potentially resulting in reduction of throughput for the TCP connections involved.

The effect of ack compression and the resulting dynamics of transport protocols under two-way traffic have been studied previously by Zhang, et al. [6], and by Wilder, et al. [7]. Zhang, et al. [6] studied TCP dynamics under two-way traffic in a datagram network by simulation, and observed that the queues in the routers exhibit periodic behavior. Wilder, et al. [7] observed a similar effect in OSI-based networks under two-way traffic causing unfairness and an overall reduction in throughput. While these studies provide a qualitative treatment of the problem, our objective in this paper is to analyze the dynamic behavior and quantify the throughput degradation of TCP connections in a two-way environment.

Some of the reduced throughput in datagram networks under two-way traffic could be attributed to the bunching of acks at the bottleneck link, behind data packets. Since the acks typically take less time to process in the routers compared to data packets, the former tend to become bunched as they travel through the network. Our interest in this paper is on TCP operating over a rate-controlled channel, such as that offered by the Available Bit Rate (ABR) service in an Asynchronous Transfer Mode (ATM) network [8]. It is expected that the burstiness of the traffic seen at the network nodes (switches or routers) would be less in a rate-controlled network as compared to one without rate control. Therefore, the effects of two-way TCP traffic introduced by the network nodes are likely to be significantly less pronounced in a rate-controlled environment. However, we show that the undesirable interaction between connections can still occur at the end systems, leading to a behavior similar to ack compression in a switch or router. Although our analysis is focussed on TCP over ABR, the results apply equally well to TCP operating over other networks providing a steady rate, predictable delay, and in-order delivery. Several earlier studies have been reported on the behavior of TCP in ATM networks [9, 10, 11], but none of them consider the effects of two-way traffic on TCP behavior.

The Available-Bit-Rate (ABR) service class [8] was defined to support delay-tolerant best-effort applications and employ rate-based feedback mechanisms to allow the sources to adjust their transmission rates to make full utilization of the available network capacity [8]. The rate-control framework developed by the ATM Forum allows a number of options for the switches to signal their congestion state to the source. The most promising of these, the *explicit-rate marking* option, is the focus of our work. With this option, the network switches compute the maximum allowable transmission rate of each connection passing through it and communicate this information to the sources in a designated field of resource-management (RM) cells transmitted periodically by the sources. The computation of the maximum rate of each connection is performed by a rate allocation algorithm within each switch. It has been shown that, by the use of a rate-allocation algorithm that maintains state, unfairness problems that occur in a datagram network due to the differences in round-trip delay of TCP connections can be avoided in an ATM network [12].

The primary objective of this paper is to analyze the dynamics of two-way TCP traffic over a rate-controlled network, develop analytical models that describe the system behavior, quantify the performance degradation, and validate the models by simulation. We show that the effect of TCP ack compression and the resulting throughput loss still persists in a rate-controlled network if the segments and acknowledgements share a common queue at the end systems. This common queue may either be at the IP layer, or at the ATM layer where the data segments of a TCP connection may share a virtual channel with the acks of a reverse connection. We consider both the symmetric case where the transmission rates of connections in both directions are identical, as well as the asymmetric case where the rates are different. In a simple configuration with one TCP connection in each direction, the connections exhibit periodic behavior and the throughput of at least one of the connections is degraded. In addition, the throughput degradation is insensitive to the relative phase of the interacting connections, but depends only on their window sizes, transmission rates, and network delays. For example, when the window size of each connection is set equal to the round-trip distance-bandwidth product, each connection is able to sustain only 66.67% of the throughput obtained under one-way traffic.

The remainder of the paper is organized as follows: In Section 2, we describe the models of the network and the end nodes assumed in our analysis, describe how ack compression occurs, and argue that TCP dynamics similar to those observed in datagram networks will cause performance degradation even while operating over the ABR service. In Section 3 we analyze the effects of two-way traffic in networks with symmetric link rates and derive expressions for throughput and maximum queue size. Section 4 extends the results to asymmetric links. We validate our analytical results with simulations in Section 5. Finally, we provide a summary of our results and directions for future work in Section 6.

2 Dynamics of TCP in a Two-Way Traffic Environment

In this section we introduce the end-system and network models assumed in our analysis and simulations, and illustrate how ack compression occurs in a two-way TCP environment. A detailed quantitative analysis of the effect of this behavior on TCP throughput will be dealt with in Section 3.

2.1 End-System and Network Models

The basic network configuration we consider has two nodes i and j communicating over a rate-controlled network providing in-order delivery, such as the Available Bit-Rate (ABR) service offered by an ATM network (Figure 2.1). The simplest configuration consists of a pair of two one-way TCP

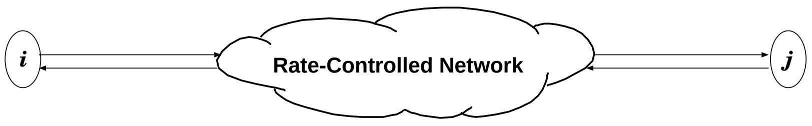

Figure 2.1: Model for 2-way traffic in a general network.

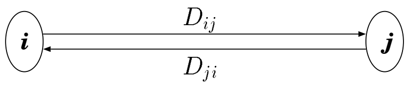

Figure 2.2: Simplified model for 2-way traffic.

connections, one transferring data from i to j and the other from j to i. Since our interest is in analyzing the end-system effects, we assume that the transmission rates of the nodes remain steady throughout the interval under observation and that the network provides a fixed-bandwidth pipe for the TCP connections between the nodes in each direction, with in-order delivery. Furthermore, in the analysis of Section 3, we assume that the network provides a constant-delay path between the end nodes. Such an assumption is realistic in the ATM ABR environment when the switches employ explicit rate allocation, since the queue sizes in the switches can be maintained small [12]. The queueing delays in network switches are taken into account in the simulation results of Section 5.

The data segments transmitted by the TCP connection in one direction share a common outgoing ATM virtual channel with the acks transmitted by the connection in the other direction. Thus, for the purpose of our analysis, the configuration in Figure 2.1 can be replaced by the simpler configuration of Figure 2.2, where the nodes are interconnected by dedicated point-to-point links, each representing a virtual channel in the actual network. Note that pairs of TCP connections between nodes that are part of a larger network can be abstracted by this model if the network maintains a constant transmission rate and delay for the connections over the interval of observation.

We refer to the TCP connection transferring data from node i to j as *connection* i and the opposite connection as *connection* j. We denote by D_{ij} the constant delay seen by TCP segments transmitted from i to j, and by D_{ji} the delay seen by segments from j to i. We also assume for simplicity that the TCP segments transmitted are of constant size. We first consider the symmetric case where the transmission rate in each direction is assumed identical, denoted by ρ in units of TCP segments/second. Later, we will extend the treatment to the case of asymmetric transmission rates.

The model of the end nodes is shown in Figure 2.3. The TCP and IP protocols operate over a rate-controlled ATM layer. We assume that appropriate segmentation and reassembly, adaptation, and transmission-rate control procedures have been implemented below the IP layer.

Two applications reside within each node, a sending application that acts as the source of data for the TCP connection originating at that node and a receiving application that acts as the sink for data arriving from the opposite node. The transmitting applications are assumed to be greedy and the receiving ones are assumed to absorb all received data immediately. Note that our analysis applies equally well when a single application in each node that both sends and receives data over a bidirectional TCP connection is assumed instead of the two applications communicating over separate TCP connections, as long as the applications are greedy.

A common process handles all TCP processing within the node. The TCP process receives data segments from the network and delivers them to the receiving application. In addition, an

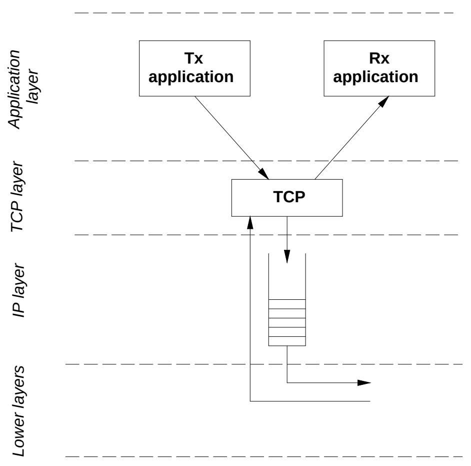

Figure 2.3: Model of an end-node in the analysis.

acknowledgement is generated and added to the outgoing queue for each segment received. The same TCP process also handles transmission of data to the opposite node by queueing one or more data segments from the transmitting application into the outgoing queue each time an ack is received from the opposite node. Finally, the TCP process is responsible for controlling the window growth of the sending TCP during the slow-start and congestion avoidance phases by incrementing it each time an ack is received, until the window reaches its maximum size.

Our analysis and simulations in this paper assume a loss-free network. If we assume that the TCP connections spend most of their time in the congestion avoidance phase, packet losses do not fundamentally alter the connection dynamics and the resulting ack compression. This would be case with TCP Reno when the packet losses are not severe enough to trigger the slow-start phase frequently. Thus, the analysis we perform here is generally applicable to environments that are throughput sensitive, where there is enough buffering to maintain a steady flow of data over a sufficiently long period so that the degradation due to ack compression is of concern.

Thus, we ignore some of the congestion control function at the TCP layer, such as recovery on a packet loss.

For simplicity, in our analysis we will assume that the TCP processing time in the end system is small and therefore, can be ignored. A detailed discussion of the effect of non-zero TCP processing time is omitted due to space constraints but can be found in [13]. The analysis and the simulation results presented in [13] validate the assumption that a non-zero TCP processing time does not alter the fundamental dynamics of two-way traffic analyzed and presented in this paper.

Segments of the forward connection and acks to the backward connection share a common FIFO queue at the IP layer that is serviced at the transmission rate of the outgoing link. This common queue is key to the occurrence of ack compression where acks to the backward connection get bunched while waiting in the queue behind data segments of the forward connection, causing the TCP dynamics we study in this paper. However, the assumption of the IP service rate being equal to transmission rate on the link is not critical if the acks of the backward connection and data segments of the forward connection share a common ATM virtual channel. A higher service rate out of the IP queue reduces queueing at the IP layer, but the same bunching effect will now occur at the ATM or adaptation layer where the data will be queued.

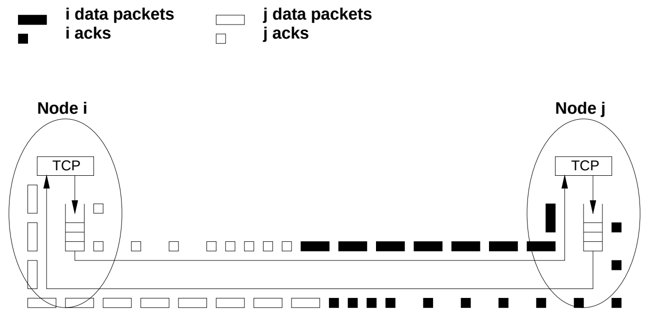

Figure 2.4: Traffic pattern in a two-way traffic configuration.

2.2 The Onset of Ack Compression

We briefly discuss now how the window-growth process during the TCP slow-start phase leads to ack compression in a two-way configuration. The result is that, on completion of the slow-start phase, each connection will always transmit an entire window of segments as a burst, followed by acknowledgements of the opposite connection.

First, we note that the sum of the window sizes of the two connections must exceed the bandwidth-delay product of the round-trip pipe for ack compression to occur. This constraint implies that the two end-systems should allow their windows to grow up to and possibly beyond the bandwidth-delay product of the path. This is indeed the likely case in a rate-controlled network, since the queueing delays within the network are likely to be small. If this condition is not satisfied, it is easy to see that no persistent queueing will occur at the IP queues at both nodes in steady state. Thus, the data segments of each connection are isolated from the acks of the other connection. Hence, if W_i and W_j are the window sizes of the two connections in segments, we must have

$$W_i + W_j > \rho(D_{ij} + D_{ji}), \tag{2.1}$$

for ack compression to occur. We will therefore assume that this condition always holds.

Before proceeding further, it is important to mention how ack compression originates at the end systems in the two-way environment and how it causes throughput degradation. The genesis of ack compression can be traced to the slow-start phase of a TCP connection that increases the window progressively at startup [1]. The slow-start algorithm sets the initial window size to one and increases it by one with every acknowledgement received. This effectively doubles the window every round-trip time. Thus, during slow start, the receipt of every ack causes the end system to add two segments to its outgoing queue. Since the outgoing queue is usually maintained in FIFO order, these two segments must be transmitted before an ack to the opposite connection can be sent. In addition, when the acks to the two transmitted segments arrive after a round-trip delay, with no data segments in between, the four data segments transmitted in response also appear back-to-back. Meanwhile, the acks of the reverse connection are queued behind the data segments, causing them to be bunched. Thus, in steady state, the entire window of each connection will be transmitted back-to-back, followed by acks of the opposite connection, as illustrated in Figure 2.4. We will later show that this behavior can persist in steady state when the windows reach their final values. A detailed description of how ack compression builds up can be found in [13].

In the preceding discussion we have merely linked the genesis of ack compression to the the slow-start window growth phase. We should make clear however that any window increase (i.e.

those occuring in the congestion avoidance phase) would have similar effects. The only difference between the two window growth phases with respect to the ack compression effects is that the size of the batch of acks will increase more rapidly in the slow-start phase as compared to the congestion avoidance phase.

It is important to realize how ack compression reduces the throughput of a TCP connection. Assume that a set of k acks returning from node i to j are bunched. Node j will generate k segments in response to these acks, which arrive back-to-back at node i. If the outgoing IP queue of node i has less than k segments at the time the segments start to arrive, node i will complete transmission of its entire IP queue before it can receive the next ack from j, causing it to be idle for a period of time, except for transmitting an ack for each segment received. This is the fundamental cause of throughput loss with two-way TCP traffic. Furthermore, if this behavior is periodic, the result is a sustained loss of throughput for one or both of the connections. In the next section we will show that such sustained loss of throughput is inevitable in a two-way configuration satisfying condition (2.1), regardless of the phase difference between the two connections at startup.

For the purpose of our analysis, the action of the TCP process can be summarized as follows: The process is invoked each time an ack arrives at the node, say node i. Processing of the ack results in a new data segment to be added to the IP queue for transmission to node j. In addition, if connection i is in its slow-start phase, its window is increased by one and an additional segment is queued in the IP queue; this latter segment is queued immediately behind the former in the IP queue, resulting in the two segments to be transmitted without an intervening ack for connection j. Furthermore, the data segments added to the IP queue in response to a bunch of acks arriving from the opposite node are transmitted as a bunch with no intervening acks. As will be shown later, this behavior causes the entire window of each connection to be transmitted always as a single bunch, giving rise to the effects we study in this paper.

The functionality assumed for the IP layer is simple. For incoming traffic the IP layer is responsible for forwarding data from the lower layer to the local TCP process. Since TCP processing time is assumed to be small, no queueing is required for incoming traffic at the IP layer. For the outgoing traffic, on the other hand, a queue may be built up at the IP layer, awaiting transmission on the link, as a result of ack bunching. Thus, in the worst case, an entire window worth of segments may be added to the outgoing queue in quick succession due to a bunch of acks received from the opposite end. Therefore, to avoid packet losses at the source node, we assume that the IP queue has a size equal to the maximum window size of the sending TCP. The interaction between TCP and IP described above is consistent with the 4.4 BSD-Lite Unix Release [14].

3 Analysis of Two-Way TCP Dynamics in Networks with Symmetric Links

In the previous section, we described how the TCP window growth during the slow-start process gives rise to the effect of ack compression under two-way traffic. In this section, we derive analytical expressions for the steady-state throughput of TCP connections in a simple network configuration with a pair of TCP flows in opposite directions after the windows have reached their maximum size. Later, in Section 5, we will validate the analytical results with extensive simulations.

Before proceeding further, we introduce the following definitions and notations. We observed in Section 2 that ack compression causes the segments of a TCP connection to be separated from the acks of the opposite connection. That is, each of the nodes transmits an entire window of segments as a single burst, followed by acks of the opposite connection. We will refer to an interval of time during which a node transmits its data segments as a *busy period* of the connection originating at the node. The busy periods for a given connection can easily be identified since these are separated by the acknowledgements of the opposite connection, as shown in Figure 2.4. We assume that the data segments transmitted are of the same size and that the transmission rates in each direction are identical, denoted by ρ in units of segments per second. The transmission time of an acknowledgment is considerably smaller than the transmission time of a data segment. Therefore, to make the analysis simple, we assume the former to be zero.

The window sizes of the two connections are assumed steady, denoted by W_i and W_j segments, respectively, for the connections originating at nodes i and j. We will denote the number of segments that are needed to fill the one-way link from i to j as L_{ij}, and that for the opposite link as L_{ji}. That is,

$$L_{ij} = \rho \cdot D_{ij}, \text{ and } L_{ji} = \rho \cdot D_{ji}.$$

Let $Q_i(t)$ denote the occupancy of the outgoing IP queue of node i at time t, considering only data segments and ignoring the space occupied by acks. Similarly, let $Q_j(t)$ denote the IP queue occupancy for node j. Furthermore, let $\tau_{i,k}$ denote the time at which the first segment transmitted by node i during the kth busy period of connection i reaches node j. Likewise we denote with $\tau_{j,k}$ the time at which the first segment transmitted by node j during the kth busy period of connection j reaches node i. Note that the number of acks that will be bunched together behind the busy period in progress at time $\tau_{i,k}$ of connection j is then given by $Q_j(\tau_{i,k})$, the number of segments in node j's outgoing queue when the first segment of the kth busy period reaches it.

Throughout this section we will assume condition (2.1), that the sum of the window sizes exceeds the bandwidth-delay product of the round-trip pipe. We build up our analysis of the effect of ack compression from a a simple case to a more complex one, to help build the understanding of the reader in steps. The simple case occurs when the window sizes and delays are such that

$$W_i > W_j + (L_{ij} + L_{ji}). \tag{3.1}$$

In this case, since the worst-case queueing delay of an ack of connection i in node j is W_j, node i never exhausts its window. Thus, each busy period of connection i will consist of W_i segments, accompanied by a sequence of W_j acks bunched together, and followed immediately by the next busy period. Since we assume that the acks are transmitted in zero time, connection i achieves perfect throughput in this case. However, the throughput of connection j is affected, since it experiences an effective round-trip delay that is equal to the transmission time of W_i segments. We will next prove these results formally.

Lemma 1: *If $W_i > W_j + (L_{ij} + L_{ji})$, then*

$$\begin{aligned} \tau_{i,k+1} &= \tau_{i,k} + W_i/\rho, \\ \text{and } \tau_{j,k+1} &= \tau_{j,k} + W_i/\rho. \end{aligned}$$

That is, the busy periods of the connections will be spaced apart by the window size W_i, exhibiting periodic behavior. Note that the above result is true in steady state, independent of the relative phase of the two interacting TCP connections at startup.

We call the ratio of the throughput of a connection to the corresponding link capacity as the *connection efficiency*, or simply *efficiency*. Since each of the connections transmits a window of segments during the transmission time of W_i segments, the efficiencies of the connections are given by

$$F_i = 1, \text{ and } F_j = W_j/W_i. \tag{3.2}$$

Proof of Lemma 1:

When the first segment of the kth busy period of connection i reaches node j, the occupancy of the IP queue of node j is $Q_j(\tau_{i,k})$. Therefore, the ack to the first segment of the busy period will leave node j at time $\tau_{i,k} + Q_j(\tau_{i,k})/\rho$ and reach node i at time t_1, where

$$t_1 = \tau_{i,k} + Q_j(\tau_{i,k})/\rho + D_{ji}. \tag{3.3}$$

We can calculate the occupancy of the outgoing IP queue in node i at time t_1 as follows: Node i started transmitting its kth busy period at time $\tau_{i,k} - D_{ij}$. Therefore, by the time the first ack to the busy period returns to node i, it would have transmitted

$$\rho\,(t_1 - (\tau_{i,k} - D_{ij})) = Q_j(\tau_{i,k}) + (L_{ij} + L_{ji}) \tag{3.4}$$

segments. Since each busy period of node i consists of W_i segments, the outgoing queue of node i must have

$$Q_i(t_1) = W_i - (Q_j(\tau_{i,k}) + L_{ij} + L_{ji}) \tag{3.5}$$

segments when the first ack of the kth busy period returns at time t_1. Note that since $Q_j(\tau_{i,k}) \leq W_j$ and $W_i > W_j + (L_{ij} + L_{ji})$, at time t_1 when the first ack to the kth busy period returns to node i the queue size will be $Q_i(t_1) > 0$. Thus, all the acks for connection j will return to node j as a bunch and connection i will always have data available in the outgoing queue for transmission. The next busy period will start at node i at time $t_1 + Q_i(t_1)/\rho$ and reach node j at time $t_1 + Q_i(t_1)/\rho + D_{ij}$. Thus,

$$\tau_{i,k+1} = t_1 + Q_i(t_1)/\rho + D_{ij}.$$

Substituting for t_1 and $Q_i(t_1)$ from equations (3.3) and (3.5), respectively, this becomes

$$\tau_{i,k+1} = \tau_{i,k} + W_i/\rho. \tag{3.6}$$

Similarly, we can show that

$$\tau_{j,k+1} = \tau_{j,k} + W_i/\rho. \tag{3.7}$$

This concludes the proof of Lemma 1.

Lemma 1 applies to the case of $W_i > W_j + (L_{ij} + L_{ji})$. The case of $W_i < W_j - (L_{ij} + L_{ji})$ is complementary, and can be handled by the same analysis. We now consider the more complex case, where the window of each connection is not large enough to overflow a pipe consisting the round-trip delay *plus* a full window of the other connection. That is,

$$W_j - (L_{ij} + L_{ji}) \leq W_i \leq W_j + (L_{ij} + L_{ji}). \tag{3.8}$$

This case covers all the instances not covered by Lemma 1, assuming Eq. (2.1) holds. In this case, we will again show that the busy periods of the connections exhibit periodic behavior, but the behavior can be more complex.

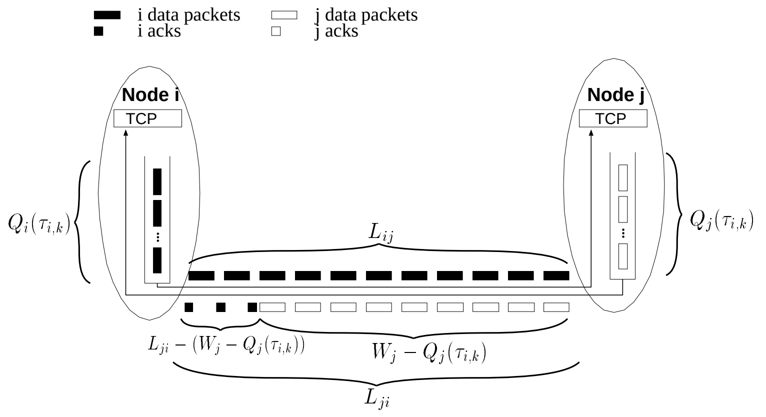

Figure 3.1: Illustration of network behavior for the proof of Lemma 2. the first segment of the kth busy period of connection i arrives at node j at time $\tau_{i,k}$ while the first segment of the mth busy period of connection j is still in transit.

Lemma 2: *Let $\tau_{i,k}$ denote the time at which the first segment of the kth busy period of connection i reaches node j, and $Q_j(\tau_{i,k})$ the queue length in node j at that time. Then,*

$$Q_j(\tau_{i,k+1}) = \min\left((W_i + W_j) - (L_{ij} + L_{ji}) - Q_j(\tau_{i,k}), W_j\right). \tag{3.9}$$

From this lemma, it is easy to observe that $Q_j(\tau_{i,k+2}) = Q_j(\tau_{i,k})$. That is, the queue behavior is periodic and the maximum queue size does not exceed $\min\left((W_i + W_j) - (L_{ij} + L_{ji}), W_j\right)$.

Proof of Lemma 2:

We need to determine $Q_j(\tau_{i,k+1})$, the queue size in node j when the $(k+1)$th busy period of connection i reaches it. Note that this queue was created by the bunched acks received by node j just before the $(k+1)$th busy period of connection i, and will be equal in number to the number of bunched acks in that ack stream. Assume that this ack stream acknowledges segments transmitted during the mth busy period of connection j.

Let $\tau_{j,m}$ denote the time at which the first segment in the mth busy period of connection j reaches node i. We can determine $\tau_{j,m}$ from $\tau_{i,k}$ as follows: At time $\tau_{i,k}$, node j had transmitted $W_j - Q_j(\tau_{i,k})$ segments of its mth busy period, as illustrated in Figure 3.1. Therefore, it would have started transmission of the mth busy period at time $\tau_{i,k} - (W_j - Q_j(\tau_{i,k}))/\rho$ and the first segment of this busy period would have reached node i after a delay of D_{ji}. Therefore,

$$\tau_{j,m} = \tau_{i,k} - (W_j - Q_j(\tau_{i,k}))/\rho + D_{ji}. \tag{3.10}$$

Let t_1 denote the time when node i completes transmission of its kth busy period. Then,

$$t_1 = \tau_{i,k} - D_{ij} + W_i/\rho. \tag{3.11}$$

During the interval $(\tau_{j,m}, t_1)$, node i is receiving segments from the mth busy period of connection j, but its outgoing queue remains non-empty, causing the acks generated by node i towards node j to be bunched. The number of bunched acks during the interval $(\tau_{j,m}, t_1)$ is given by $\min(\rho(t_1 - \tau_{j,m}), W_j)$. Using Eq. (3.10) and (3.11),

$$\rho(t_1 - \tau_{j,m}) = (W_i + W_j) - (D_{ij} + D_{ji})\rho - Q_j(\tau_{i,k}). \tag{3.12}$$

Therefore, the queue size seen by the first segment of the $(k+1)$th busy period of connection i is

$$\begin{aligned} Q_j(\tau_{i,k+1}) &= \min\left((W_i + W_j) - (D_{ij} + D_{ji})\rho - Q_j(\tau_{i,k}), W_j\right) \\ &= \min\left((W_i + W_j) - (L_{ij} + L_{ji}) - Q_j(\tau_{i,k}), W_j\right). \end{aligned} \tag{3.13}$$

This concludes the proof of Lemma 2.

Although Lemma 2 establishes the periodic nature of the queue behavior in the nodes, it does not provide us an expression for the period. We can now use this result to calculate the interval between busy periods transmitted by a node, and thus the efficiencies of the connections. The following lemma allows us to quantify the effect of ack compression on throughput.

Lemma 3: *If the window sizes and delays are such that* $W_j - (L_{ij} + L_{ji}) \leq W_i \leq W_j + (L_{ij} + L_{ji})$, *then*

$$\tau_{i,k+2} - \tau_{i,k} = (W_i + W_j)/\rho + (D_{ij} + D_{ji}). \tag{3.14}$$

That is, each connection gets to transmit two consecutive busy periods within an interval of $(W_i + W_j)/\rho + (D_{ij} + D_{ji})$. Thus, the efficiencies of the connections are given by

$$F_i = \frac{2W_i}{(W_i + W_j) + (L_{ij} + L_{ji})}, \quad \text{and } F_j = \frac{2W_j}{(W_i + W_j) + (L_{ij} + L_{ji})}.$$

Proof of Lemma 3:

Let t_1 denote the time at which the first ack to the kth busy period of connection i returns to node i. Then,

$$t_1 = \tau_{i,k} + Q_j(\tau_{i,k})/\rho + D_{ji}. \tag{3.15}$$

When this ack reaches node i, the outgoing queue of i may or may not be empty. We consider these two cases separately below:

Case 1: $Q_i(t_1) > 0$. This case occurs when node i has not completed transmission of segments in its kth busy period at time t_1. That is,

$$W_i/\rho > t_1 - (\tau_{i,k} - D_{ij}).$$

Substituting for t_1 from Eq. (3.15), this condition becomes

$$W_i > (L_{ij} + L_{ji}) + Q_j(\tau_{i,k}). \tag{3.16}$$

In this case, node i will follow transmission of the kth busy period with compressed acks for an entire window W_j of connection j, followed by the $(k+1)$th busy period of connection i. Since we ignore the transmission time of acks, node i will begin transmission of the $(k+1)$th busy period at time $\tau_{i,k} + W_i/\rho - D_{ij}$, the first segment of this busy period will reach node j at $\tau_{i,k+1} = \tau_{i,k} + W_i/\rho$, and the last bit of the last packet transmitted in the $(k+1)$th busy period will depart from node i at time $t_1' = \tau_{i,k} + 2(W_i/\rho) - D_{ij}$.

Since the $(k+1)$th busy period follows compressed acks for W_j segments of connection j, the first bit of this busy period will see a queue size of $Q_j(\tau_{i,k+1}) = W_j$ segments in node j. Note that the same result can be obtained by combining Eq. (3.13) and Eq. (3.16). Therefore, the first ack to the $(k+1)$th busy period will leave at time $\tau_{i,k+1} + W_j/\rho$ and will arrive at node i at time $t_2 = \tau_{i,k+1} + W_j/\rho + D_{ji}$. Since $W_i \leq W_j + (L_{ij} + L_{ji})$, we can easily conclude that $t_2 \geq t_1'$ and therefore, node i would have completed transmission of the $(k+1)$th busy period of connection i

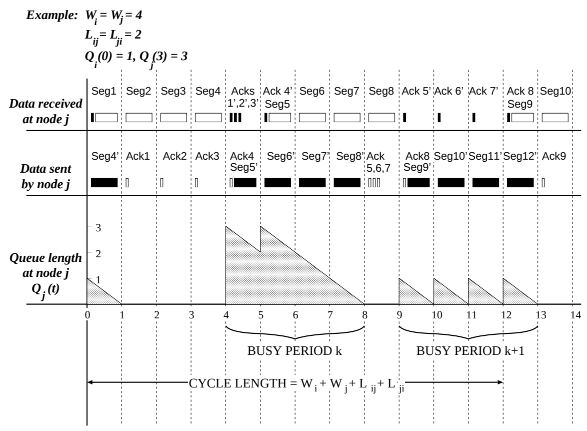

Figure 3.2: Example behavior of node j in a two-way TCP configuration.

by then. Thus, the $(k+2)$th busy period will begin at time t_2 and arrive at node j after a delay of D_{ij}. Therefore,

$$\begin{aligned}\tau_{i,k+2} &= t_2 + D_{ij} \\ &= \tau_{i,k+1} + W_j/\rho + D_{ji} + D_{ij} \\ &= \tau_{i,k} + (W_i + W_j)/\rho + (D_{ij} + D_{ji}).\end{aligned}$$

Case 2: $Q_i(t_1) = 0$. This case can occur only if $W_i \leq W_j + (L_{ij} + L_{ji})$. In this case the $(k+1)$th busy period of connection i will start at node i at time t_1, and its first segment will reach node j at time $t_1 + D_{ij}$. Hence,

$$\tau_{i,k+1} = t_1 + D_{ij} = \tau_{i,k} + Q_j(\tau_{i,k})/\rho + (D_{ij} + D_{ji}). \tag{3.17}$$

The first ack to this busy period will leave node j at time $\tau_{i,k+1} + Q_j(\tau_{i,k+1})/\rho$ and will reach node i at time

$$t_3 = \tau_{i,k+1} + Q_j(\tau_{i,k+1})/\rho + D_{ji}. \tag{3.18}$$

Substituting for $\tau_{i,k+1}$ from Eq. (3.17) and for $Q_j(\tau_{i,k+1})$ from Eq. (3.13), this becomes

$$t_3 = \tau_{i,k} + (W_i + W_j)/\rho + D_{ji}. \tag{3.19}$$

Node i would have transmitted its entire $(k+1)$th busy period at time t_3, so the $(k+2)$th busy period begins at t_3. The first segment of this busy period reaches node j at time $t_3 + D_{ij}$. Therefore,

$$\begin{aligned}\tau_{i,k+2} &= t_3 + D_{ij} \\ &= \tau_{i,k} + (W_i + W_j)/\rho + (D_{ij} + D_{ji}).\end{aligned} \tag{3.20}$$

This concludes the proof of Lemma 3.

The example in Figure 3.2 illustrates the periodic behavior of the TCP connections for the case $W_j - (L_{ij} + L_{ji}) \leq W_i \leq W_j + (L_{ij} + L_{ji})$. In this case, the window sizes are taken as $W_i = W_j = 4$ segments, and the time to transmit each segment on the link as one unit. The network delay in each direction is assumed as 2 units, that is $L_{ij} = L_{ji} = 2$. Thus, a window size of 4 segments allows each connection to achieve its maximum throughput in a one-way configuration. We will show that the throughput is substantially reduced in a two-way configuration because of the interaction between the connections.

Figure 3.2 illustrates the operation of node j during an interval of time. The segments transmitted by connection i are marked as $Seg1, Seg2, Seg3, \ldots$ in the figure, and those transmitted by connection j as $Seg1', Seg2', Seg3', \ldots$. At time 0, node j is transmitting the last of the four segments constituting the $(k-1)$th busy period of connection j. This segment is labeled as $Seg4'$ in the figure. At the same time, node j is receiving the first segment of the mth busy period of the opposite connection. The last segment of this mth busy period is completely received by time $t = 4$. Note that each of the acks to segments of this busy period leaves node j immediately without any ack compression.

The acks to the $(k-1)$th busy period of connection j starts arriving at $t = 4$. The acks to the first three segments arrive bunched at time $t = 4$, while the ack to the last segment is received in the next time-slot. When the first three acks are received, a queue of three segments is immediately built up at node j. By time $t = 5$, the first segment of this queue is transmitted, and a new segment is added in response to ack $4'$, maintaining the queue size at 3 segments. The queue is completely drained by time $t = 8$. Note that the segments transmitted from $t = 4$ to $t = 8$ consists of the kth busy period of connection j.

The arrival of the $(m+1)$th busy period of connection i at node j occurs at time $t = 5$ and overlaps with the transmission of the kth busy period of connection j. Therefore, the acks to the first three segments of the former are queued behind the segments of the latter, and transmitted as a bunch at time $t = 8$ (acks 5, 6, 7 in figure).

The acks to segments of the kth busy period of connection j starts to arrive at time $t = 9$ with no bunching. As a result, node j is receiving one ack in each of the time-slots 9, 10, 11, 12 and transmitting a segment in response. These transmitted segments constitute the $(k+1)$th busy period of connection j. It is easy to observe that the state of the system at $t = 12$ is identical to that at time $t = 0$, so the behavior will repeat indefinitely.

Note that during time-slots 1, 2, 3 and 8, node j is transmitting only acks, but no segments, leaving its outgoing link virtually unused. Thus, the efficiency of connection j is only $8/12 = 66.67$ percent. Similarly, it can be seen that the efficiency of the opposite connection is also 66.67 percent. It is important to note that the queue lengths at node j in this example will change if the relative phase of the two connections is changed, but both the periodic behavior and the efficiency will remain unchanged.

The following theorem summarizes the efficiencies of TCP connections under two-way traffic for all ranges of window sizes, under the condition that $W_i + W_j > L_{ij} + L_{ji}$.

Theorem 1: *The efficiency F_i of connection i in a two-way traffic configuration with symmetric link rates is given by*

$$F_i = \begin{cases} 1, & \text{if } W_i > W_j + (L_{ij} + L_{ji}); \\ \frac{2W_i}{(W_i+W_j)+(L_{ij}+L_{ji})}, & \text{if } W_j - (L_{ij} + L_{ji}) \le W_i \le W_j + (L_{ij} + L_{ji}); \\ \frac{W_i}{W_j}, & \text{otherwise.} \end{cases} \qquad (3.21)$$

The proof of this theorem follows directly from Lemmas 1, 2, and 3. Note that the third part is complementary to the first, and is covered by Lemma 1. The theorem asserts that, under two-way traffic, the throughput of one or both connections will always be degraded, as long as the sum of the window sizes is larger than the round-trip delay-bandwidth product. The connection with the larger window always receives a higher proportion of the attainable throughput. Furthermore, in the extreme case where there propagation delays are small compared to the window sizes (as may be

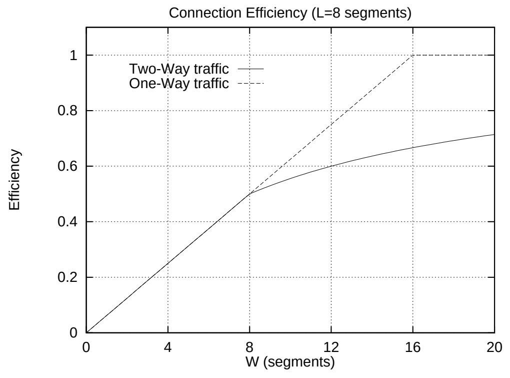

Figure 3.3: Effect of window size on connection throughput for $L = 8$ segments.

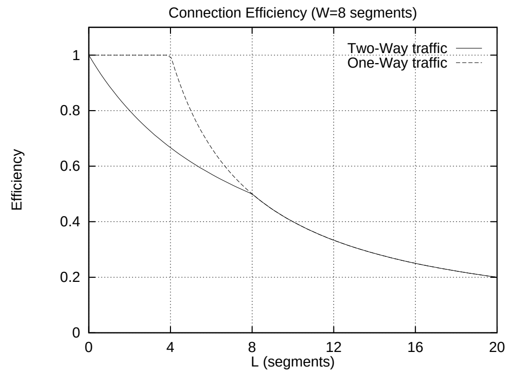

Figure 3.4: Effect of link delay on connection throughput for $W = 8$ segments.

the case in a local-area network), differences in the window-sizes of the two connections will result in throughput unfairness. Another important conclusion that can be drawn from Eq. (3.21) is that, under two-way traffic, the throughput of one connection cannot be increased without degrading that of the other. The throughput of both connections, however, can be increased by increasing their window sizes. To make this point clear, consider a simple case where the windows of both connections are of equal size and the propagation delays in both directions are also equal, that is, $W_i = W_j = W$ and $L_{ij} = L_{ji} = L$. In this case the efficiency F for each of the connections will be

$$F = \frac{W}{W + L},$$

as compared to $\min(W/2L, 1)$ with one-way traffic. If the window size is just large enough to fill the round-trip network pipe, that is if $W = 2L$, the efficiency for each connection will be only 66.67%. Increasing the window size of both connections beyond this point improves their throughput, but has the undesirable consequence of large queues at the end systems.

Figures 3.3 and 3.4 further illustrate the effect of window sizes and network delays on TCP throughput under two-way traffic. Here we assume equal windows and identical delay-bandwidth products in each direction. Figure 3.3 shows the variation of connection efficiency with the window size, keeping the one-way delay at $L = 8$ segments. The throughput increases linearly until $W = L$, that is until 50% efficiency. While the efficiency continues to increase linearly beyond this point in a one-way configuration, under two-way traffic the throughput increases much slower, reaching 66.67% of the maximum at a window size of $2L$. Figure 3.4 shows the same effect, this time plotting the connection efficiency versus the link delay-bandwidth product, keeping the window size constant. The throughput degradation occurs in the region of $L = 0$ to 8, and again illustrates the basic result that window increases beyond $W = L$ provide only much smaller throughput improvements than in the one-way case.

We conclude this section with an expression for the maximum size of the outgoing IP queue in each node under two-way traffic. It is easy to see that the maximum queue size in node j is seen when the first segment of a busy period of connection i reaches node j. In addition, the queue size can never exceed the window size of the connection originating at the node. Thus, we can state the following lemma.

Lemma 4: *In steady state, the maximum occupancy $Q_{i,max}$ of the IP queue of node i under two-way traffic is given by*

$$Q_{i,max} \leq \begin{cases} W_i - (L_{ij} + L_{ji}), & \text{if } W_i > W_j + (L_{ij} + L_{ji}); \\ (W_i + W_j) - (L_{ij} + L_{ji}), & \text{if } W_j - (L_{ij} + L_{ji}) \leq W_i \leq W_j + (L_{ij} + L_{ji}); \\ W_i, & \text{otherwise.} \end{cases} \tag{3.22}$$

Proof: The second case follows directly from Lemma 2, and the third case from the fact that the queue size can never exceed the window size of the connection originating at the node. Now consider the first case. The maximum queue size in node i is seen when the first segment of a busy period of connection j reaches it. Let $\tau_{j,m}$ denote the time at which the first segment of the mth busy period of connection j reaches node i. The queue size at this time can be found by noting that the number of queued segments in node i, plus the number of segments and acks belonging to connection i in the rest of the network, must equal its window size W_i.

The first segment of the mth busy period of connection j would have been transmitted by node j at time $t_1 = \tau_{j,m} - D_{ji}$. At time t_1, the outgoing queue of node j contained no acks of connection i. During the interval $(t_1, \tau_{j,m})$, a total of $(\tau_{j,m}, t_1)\rho = D_{ji}\rho = L_{ji}$ segments from connection i arrived at node j. In addition, L_{ij} segments of connection i are in transit on the link from i to j at time $\tau_{j,m}$. Therefore, the total number of segments and acks of connection i in the entire system at time $\tau_{j,m}$ is given by

$$Q_i(\tau_{j,m}) + L_{ij} + L_{ji}.$$

Equating this to the window size W_i, we get

$$Q_i(\tau_{j,m}) = W_i - (L_{ij} + L_{ji}).$$

This concludes the proof of Lemma 4.

4 Analysis for Networks with Asymmetric Links

We can now extend the results of Section 3 to the more general asymmetric case where the transmission rates in opposite directions are different. Let ρ_i and ρ_j denote the transmission rates of connections i and j, respectively. The condition for occurrence of ack compression then becomes

$$\frac{W_i}{\rho_i} + \frac{W_j}{\rho_j} > D_{ij} + D_{ji}. \tag{4.1}$$

That is, the sum of the transmission times of the windows of the two connections must be larger than the round-trip delay for ack compression to occur. When this necessary condition is satisfied, we can generalize Theorem 1 for connection throughput in the asymmetric case as follows:

Theorem 2: *The efficiency F_i of connection i in a two-way traffic configuration with asymmetric link rates is given by*

$$F_i = \begin{cases} 1, & \text{if } W_i/\rho_i > W_j/\rho_j + (D_{ij} + D_{ji}); \\ \frac{2(W_i/\rho_i)}{(W_i/\rho_i + W_j/\rho_j) + (D_{ij} + D_{ji})}, & \text{if } W_j/\rho_j - (D_{ij} + D_{ji}) \leq W_i/\rho_i \leq W_j/\rho_j + (D_{ij} + D_{ji}); \\ \frac{(W_i/\rho_i)}{(W_j/\rho_j)}, & \text{otherwise.} \end{cases} \tag{4.2}$$

The proof of this theorem is similar to that of Theorem 1 and is therefore omitted. We can similarly determine the maximum queue sizes at the end nodes by extending Lemma 4.

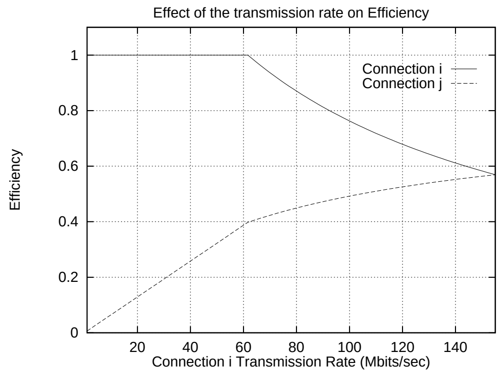

Figure 4.1: Effect of transmission rate ρ_i of connection i on the efficiencies of connections i and j in an asymmetric configuration. The transmission rate of connection j is held constant at 155 Mbits/sec. ($W_i = W_j = 128$ Kbytes, $D_{ij} = D_{ji} = 5$ msecs).

Lemma 5: *In steady state, the maximum occupancy $Q_{i,max}$ of the IP queue of node i under two-way traffic with asymmetric link rates is given by*

$$Q_{i,max} \leq \begin{cases} W_i - (D_{ij} + D_{ji})\rho_i, & \text{if } W_i/\rho_i > W_j/\rho_j + (D_{ij} + D_{ji}); \\ \left(\frac{W_i}{\rho_i} + \frac{W_j}{\rho_j} - D_{ij} - D_{ji}\right)\rho_i, & \text{if } W_j/\rho_j - (D_{ij} + D_{ji}) \leq W_i/\rho_i \leq W_j/\rho_j + (D_{ij} + D_{ji}); \\ W_i, & \text{otherwise.} \end{cases} \tag{4.3}$$

Eq. (4.2) suggests that, if the transmission rates in the two directions differ significantly, then the efficiency of the high-bandwidth connection will be limited by the window transmission time of the lower-bandwidth connection. This could lead to significant performance degradation. To illustrate this problem, consider an example where the window sizes are chosen as $W_i = W_j = 128$ Kbytes and the link delays are $D_{ij} = D_{ji} = 5$ milliseconds. Assume that connection j is allocated a steady rate of 155 Mbits/second. Figure 4.1 shows the variation of the efficiencies of connections i and j as a function of the rate allocated to connection i. When the bandwidth available to connection i is low, connection j is able to use only a small percentage of its available link bandwidth. For example, when ρ_i is 20 Mbits/second, the efficiency of connection j is below 15%. As the bandwidth of connection i is increased, connection j's throughput increases linearly until about 60 Mbits/second, while connection i's throughput remains at its ideal value. Case 3 of Eq. (4.2) holds in this region for connection j. Beyond this point, the throughput is determined by case 2 of the equation, and the throughput increase of connection j occurs at a lower rate. For example, increasing connection i's transmission rate from 60 to 155 Mbits/second increases the effective efficiency of connection j to go from 40% to 60%. But what is disappointing is that increasing the transmission rate of connection i doesn't automatically result in a benefit to it. The efficiency of connection i goes down to 60%, while the efficiency of connection j went from 40% to 60%. In addition,the total increase of 95 Mbits/second in link bandwidth results in a sum total increase in throughput of the two connection by only 64 Mbits/second. The remaining 31 Mbits/second is lost due to the increasing effects of ack-compression. Thus, the throughput degradation due to two-way traffic can be even more dramatic with asymmetric link rates as compared to the symmetric case.

5 Model Validation

In this section we validate the analytical results derived in the two previous sections by simulation using several network configurations. In order to simplify the analysis we made several simplifying assumptions in the last sections such as TCP acknowledgments were considered to have zero transmission time. In the simulations the acknowledgments have a small but non-zero size. We will now show that these assumptions do not significantly affect the results. Finally, we show that the results are applicable to more general network topologies than the simple two-node configuration considered in the last sections, by simulating more complex network configurations.

5.1 Simulation Model

We now provide an overview of the simulation models used in this section. More detailed descriptions of specific topologies used in the simulations will be given in the corresponding part of the section describing the simulation results.

The simulations were performed using the OPNET simulation tool. The links in the network are full-duplex with a capacity of 155 Mbits/second each, unless otherwise specified. The switches are nonblocking, output-buffered crossbars. There is one queue per output port for ABR traffic and its scheduling policy is FIFO, with each output queue being shared by all the virtual channels (VCs) sharing the outgoing link. The switches support the explicit rate allocation algorithm of Kalampoukas, et al. [12]. Each of the nodes implements the ABR source policy defined by ATM Forum [15]. The parameters for the source policy were chosen as follows: The interval between transmission of Resource Management (RM) cells was set at $Nrm = 32$ cells, the Rate Increase Factor (RIF) was chosen as 1/64, and the Initial Cell Rate (ICR) was selected as 1/50 of the Peak Cell Rate (PCR). Since we are interested in steady-state performance, and the rate-allocation algorithm converges to a stable and fair allocation in steady state, other parameters of the source policy are not critical to these simulations.

At the source, we implemented models of ATM Adaptation Layer Type 5 (AAL 5), IP, and TCP. The AAL performs segmentation and re-assembly between IP packets and ATM cells. Each IP packet is extended by eight bytes to accommodate the AAL header. Unless stated otherwise, the service rate of the outgoing IP queue is set at the transmission rate of the connection at the ATM layer. The model for TCP used in the simulations is based on the TCP-Reno version. The TCP segment size was set at 9180 bytes in all the simulations.

5.2 Simulation Results for a Simple Network with Symmetric Links

We consider first the simple configuration shown in Figure 5.1, consisting of two nodes and a pair of TCP connections set up between them, one in each direction. The connections originating at the left and right nodes are designated as 1L and 1R, respectively, in the figure. The link capacity in each direction is identical and equal to 155 Mbits/second. After taking into account all the overhead at the lower layers, including the bandwidth lost due to RM cells, the bandwidth available to each TCP connection is approximately 85% of the link capacity, that is about 131 Mbits/second. In our simulation results, the measured efficiency for a connection is normalized to this maximum throughput attainable by a TCP connection.

The one-way propagation delay in this configuration is 8 msecs. This is approximately equal to the transmission time of 14 segments. However, the fragmentation of a segment to cells at the end system causes an additional delay equivalent to the transmission time of one segment. Therefore, the effective one-way delay between end systems, expressed in number of segments, is about 15. The window size for both the connections is set to 256 Kbytes, equivalent to 28 segments.

Initially we assume that both the connections open concurrently at time $t = 0$. Since the behavior of the connections will be symmetric in this case, we have plotted the simulations only for the left-to-right connection. The transmission rate available to each connection at the ATM layer, designated as the Allowed Cell Rate (ACR), starts at the initial rate of about 3.5 Mbits/second and increases gradually to its maximum value, as shown in Figure 5.2. The ACR reaches its steady-state value in about 110 msecs, and remains at this value for the rest of the simulation time, because of the rate allocation algorithm. Note that the rate increase process is not smooth but consists of a sequence of steps. This is because of the gaps in the data flow during the TCP slow-start process, and has been analyzed in [16].

The progress for the left-to-right TCP connection is shown in Figure 5.4, in terms of the increase in TCP sequence numbers transmitted by the connection. Note that these are the sequence numbers of the packets as they exit the TCP layer, not when they exit the outgoing IP transmission queue. Also shown in Figure 5.4 for comparison is the slope of the optimal sequence number growth that would have occurred if the two-way traffic effects were absent. Thus, the difference in slope between the optimal and the measured sequence number growth plots is a direct measure of the performance degradation caused by two-way traffic. Except for the initial start-up period during which the transmission rate and the congestion window size are still increasing, the progress is steady. However, notice the step-like increase of the TCP sequence numbers. These jumps are caused when a cluster of bunched acknowledgments arrive at the source and cause a corresponding number of segments to be added to the outgoing queue. The queue size at the end-systems exhibits oscillatory behavior, as observed in the analysis of Section 3. Figure 5.3 shows the queue-size behavior observed in the simulation. Because the two connections open simultaneously, the queue size reached at the beginning of each busy period is identical, about 13 segments. The solid black areas shown at the peaks of Figure 5.3 signify the busy periods of the connection, and are caused by the high-frequency transitions occurring from the addition of new segments into the outgoing queue in response to the arrival of acks. The analytical result of Eq. (3.9) asserts that, if the queue size is 13 segments during a given busy period, the queue size during the next busy period will be be $W_i + W_j - L_{ij} - L_{ji} - 13 = 28 + 28 - 15 - 15 - 13 = 13$, which is exactly what we observe in the simulation results. Considering that in the analysis we have ignored the transmission time of acknowledgements and the fact that packet arrivals and departures may not be synchronized, we conclude that there is a very good match between the simulation results and the analysis with respect to the IP queue size at the end-systems.

Since $W_i = W_j = 28$, and $L_{ij} = L_{ji} = 15$, the expected efficiency F_i for the left-to-right connection, from analysis, will be

$$F_i = \frac{28}{28 + 15} \approx 65.1\%.$$

The actual throughput observed in the simulation is in close agreement with this analytical result, as may be observed from the sequence-number plot of Figure 5.4.

We have also studied the TCP connection behavior when the two connections open in a staggered fashion. We have verified that the dynamics are the same as with the case that the connections are

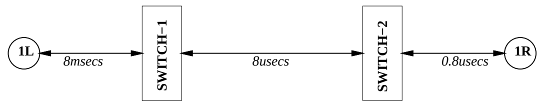

Figure 5.1: Two-Way traffic configuration with a single TCP connection in each direction.

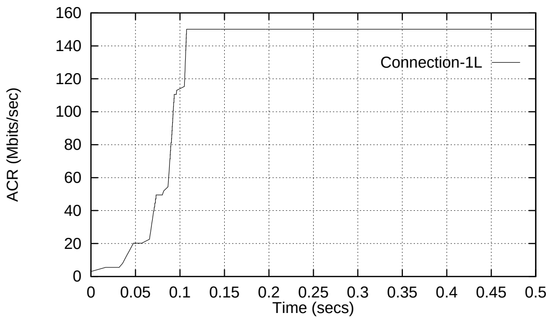

Figure 5.2: Allowed Cell Rate (ACR) at the ATM layer for the left-to-right connection.

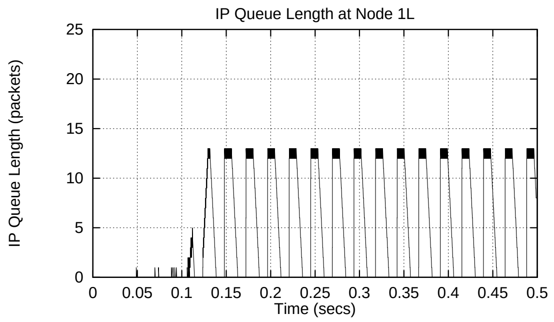

Figure 5.3: IP queue size at node 1L (data packets only).

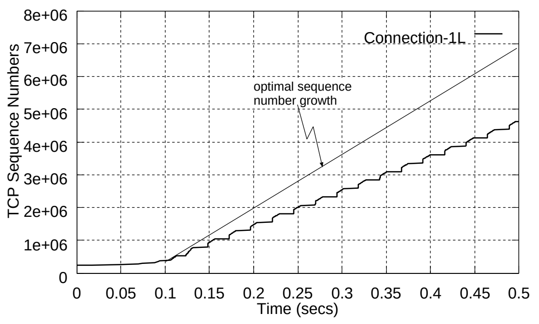

Figure 5.4: Sequence number growth for the left-to-right connection.

synchronized and that the queue behavior and connection efficiencies are accurately modeled by the presented analysis. A detailed discusion and simulation results of this case is omitted due to space contraints and can be found in [13].

5.3 Simulation Results for Networks with Asymmetric Links

We now proceed to verify our analytical results for the asymmetric case where the transmission rates in the two directions differ. We consider again the simple configuration of Figure 5.1. However, the transmission rate for the left-to-right connection is now set to 51.6 Mbits/second, while the transmission rate for the right-to-left direction is set at 155 Mbits/second. The maximum window size for the left-to-right connection is set to 14 segments and for the right-to-left connection to 28 segments. The link propagation delays remain unchanged.

From Eq. (4.2), we can estimate the efficiency for the faster right-to-left connection j as

$$F_i = \frac{2W_j}{\left(\frac{W_i}{\rho_i} + \frac{W_j}{\rho_j} + D_{ij} + D_{ji}\right)\rho_i} \approx 56\%.$$

The efficiency for the slower left-to-right connection is higher, about 80%. Figures 5.5 through 5.8 summarize the simulation results. Figure 5.5 shows that the throughput for the right-to-left connection 1R is substantially impacted by the effect of two-way traffic, where the lower bandwidth

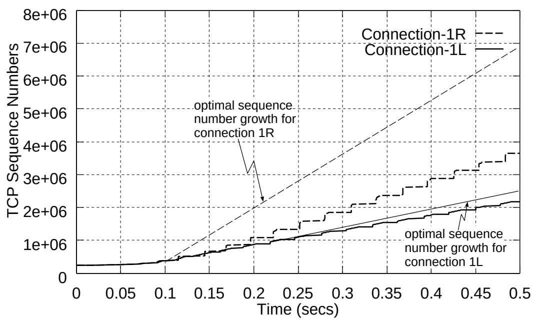

Figure 5.5: Sequence number growth for the left-to-right and the right-to-left connections.

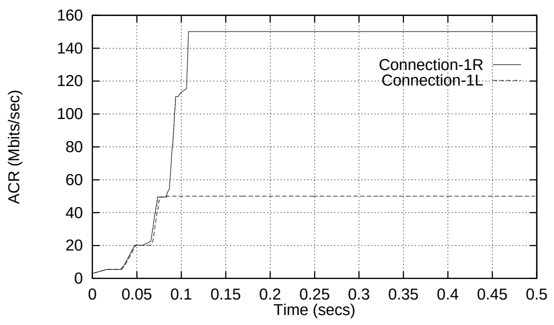

Figure 5.6: Allowed Cell Rate (ACR) at the ATM layer for the two connections.

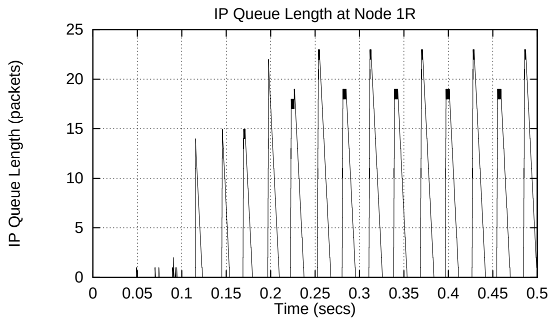

Figure 5.7: IP queue size at node 1L (data packets).

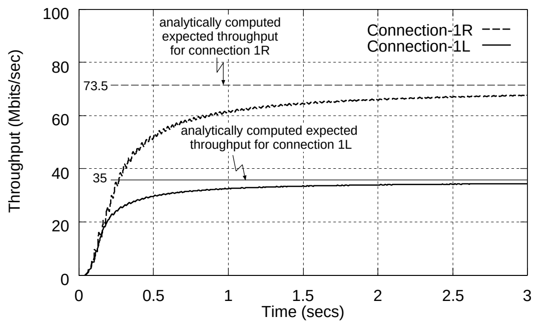

Figure 5.8: Connection throughput averaged from time $t = 0$ and comparison with the analytically computed expected throughput.

available to the connection in the opposite direction results in substantial queueing of its acknowledgements. Note that the throughput of the connections, shown averaged in Figure 5.8 from the start of simulation at time 0, is close to that predicted by Eq. (4.2). The right-to-left connection reaches a steady-state throughput of 73.5 Mbps, which is virtually identical to the analytically predicted throughput based on the efficiency of 56%. This throughput degradation is also visible in the IP queue behavior at the right node, shown in Figure 5.7, where the interval between peaks is much longer as compared to the queue-size behavior in Figure 5.3. Note also that there is a increase in the idle time on the channel as observed by the differences in the two figures, where the space between the cycles increases: this leads to a corresponding reduction in the overall efficiency.

5.4 Simulation Results with Multiple Connections Between End Systems

Up to now, we considered configurations where there is only a single TCP connection set up between each pair of end nodes. In this section we examine the case when there are multiple TCP connections set up between the same node pairs in each direction. In this case the behavior of the system depends on the relative phase of the TCP connections, and obtaining a general model is difficult. However, there are some subcases where the analytical results derived in Sections 3 and 4 can be readily applied.

For simplicity we consider first the case when all the TCP connections in each direction open simultaneously. In this case, we can aggregate the TCP connections in each direction and replace them with a single connection whose window size and transmission rate are the sum of the corresponding parameters of the constituent connections. We will refer to this equivalent connection as the *fat* connection. All the analytical results derived in Sections 3 and 4 can now be applied to the fat connections. The system will go through a series of busy and idle periods. During a busy period of a fat connection, the connection transmits an amount of data equal to the sum of the window sizes of the constituent connections. Furthermore, the data segments and acknowledgments for the set of connections flowing in the same direction will appear as separate clusters, with no interleaving between them.

We can relax the assumption of simultaneous opening of connections slightly by noting that, as long as a new connection transmits its first segment while the current fat connection is going through its busy period, the separation between data segments and acks is preserved. Therefore, the analytical model can still be applied to the fat connection if this condition holds. If we denote by I the set of connections from left to right, and by J the set of connections in the opposite direction in a two-node configuration, then the efficiency of a given connection $l \in I$ will be

$$\begin{aligned} F_l &= \frac{2W_l}{\sum_{i\in I} W_i + \sum_{j\in J} W_j + L_{ij} + L_{ji}} \cdot \frac{2\sum_{i\in I} W_i}{2W_l} \\ &= \frac{2\sum_{i\in I} W_i}{\sum_{i\in I} W_i + \sum_{j\in J} W_j + L_{ij} + L_{ji}}. \end{aligned}$$

To study the behavior of two-way TCP traffic with multiple TCP connections in each direction, we simulated the configuration shown in Figure 5.9, designated as the R-1 configuration. the network consists of three nodes on the left connected to three nodes on the right through a path consisting of two switches. A set of four TCP connections are set up in each direction between every node on the left and the corresponding node on the right. Thus, there are a total of 12 connections in each direction. The link capacities are 155 Mbits/second each and the link propagation delays are as shown. Note that the delay for connections originating or terminating at nodes 1L and 2L is about three orders of magnitude higher than that seen by the connections between nodes 3L and 3R. Note that the ack-compression continues to occur even when the RTT for the connections is so dramatically different.

The maximum aggregate throughput attainable by each set of TCP connections originating at a node is approximately $131/3 \approx 43.6$ Mbits/second. Since the multiplexing of the outgoing traffic takes place at the node's IP queue, we set the processing capacity for the IP process also to be 43.6 Mbits/sec. The maximum window size for each connection is set to 64 Kbytes, equivalent to 7 packets. Therefore, the aggregate window for the equivalent fat connection will be 256 Kbytes. The one-way delay for the fat connections originating or terminating at nodes 1L and 2L is approximately equal to the transmission time of 9 segments, while the delay is negligible for the path between nodes 3L and 3R.

The simulation results for this configuration are shown in Figures 5.10 through 5.13. In Figure 5.10 we observe again the effect of ack compression causing the sequence numbers to increase in periodic steps for the TCP connections originating at node 1L. Since the connection efficiency decreases with an increase in delay, the short connections originating at node 3L will likely have

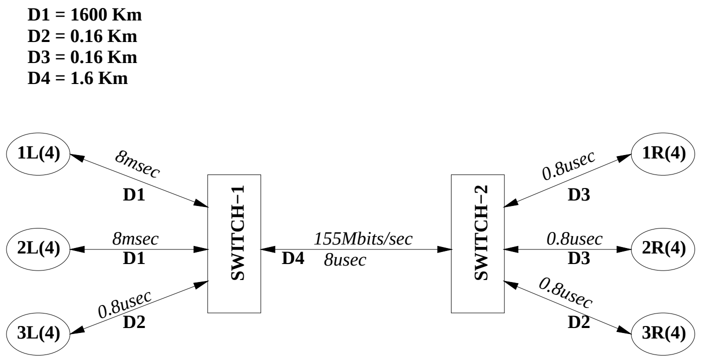

Figure 5.9: R-1 Configuration. A set of four TCP connections are set up between nodes with same index number in each direction.

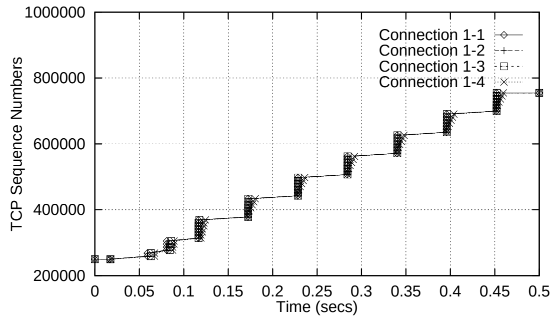

Figure 5.10: TCP sequence number growth for the four connections originating at node 1L.

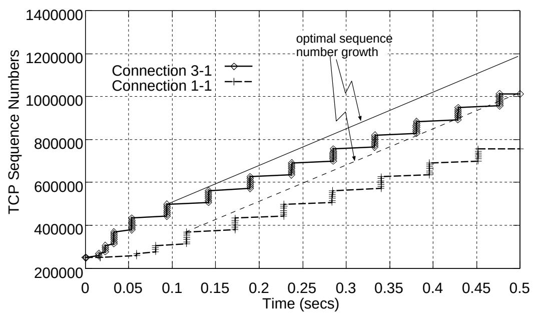

Figure 5.11: TCP sequence number growth of one connection originating at node 1L compared with that of a connection originating at 3L.

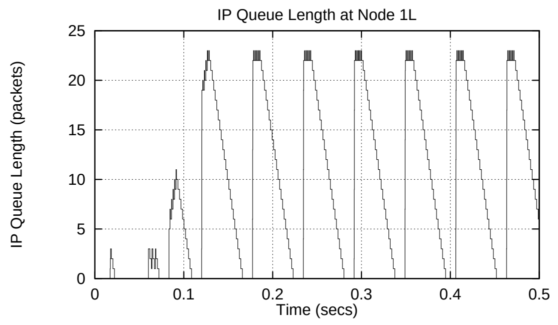

Figure 5.12: IP Queue Size at node 1L (due to data packets).

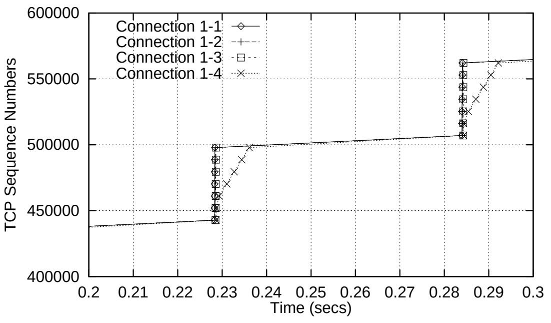

Figure 5.13: TCP sequence number growth for the four connections originating from node 1L: a magnified view.

higher throughout as compared to the connections originating at nodes 1L and 2L. The sequence number plots in Figure 5.11 confirm this behavior. In addition, the IP queue behavior at node 1L, shown Figure 5.12, confirms that the aggregated traffic from the four TCP connections originating at node 1L behave as a fat connection with a window equal to the sum of the windows of the four constituent connections.

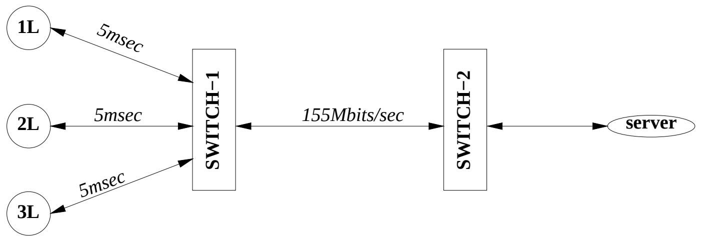

Figure 5.14: A client-server configuration. A pair of TCP connections, one in each direction, is set up between the server node and each of the remote nodes.

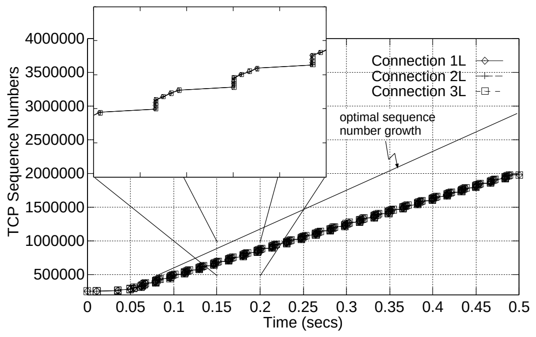

Figure 5.15: Sequence number growth for the three TCP connections flowing from the server node to the clients.

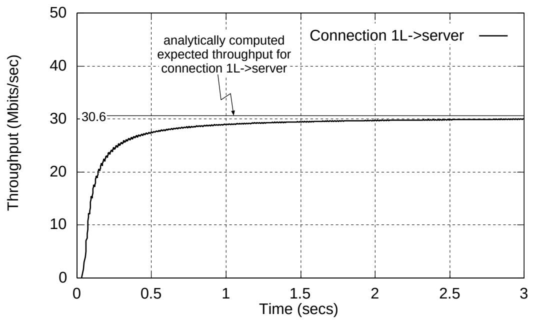

Figure 5.16: Throughput for the connection from node 1L to server, averaged from time 0.

Figure 5.13 presents a detailed view of the sequence number growth shown in Figure 5.10. An interesting observation we can make in Figure 5.13 is that some of the segments for the fourth connection originating at node 1L are spaced apart evenly by the transmission time of one segment. This behavior can be explained by observing that the IP queue in node 1L oscillated between 0 and 23 segments. Note that the sequence number plots are based on the time a given segment is released by the corresponding TCP process, and not on the time at which the packet is actually serviced from the IP queue. Thus, a total of 23 acknowledgments arrive at node 1L in a cluster, causing the addition 23 new segments to the IP queue. Since the window size for each connection is 7 packets, and because the four connections are synchronized, the ack stream for the fat connection originating at node 1L consists of a total of 28 acks, out of which 23 arrive in a cluster and the remaining spaced apart by one segment transmission time. In this case, the last acks arriving at a spacing of one segment belongs to the fourth connection, causing new segments to be added to the IP queue at intervals of one segment transmission time.

A topic that deserves further investigation is the mapping of TCP connections set up between the same end systems to ATM connections that transport the data through the ATM network. The approach currently being undertaken is to map all TCP connections set up between the end-systems (in general, subnetworks containing the end-systems) to a single ATM connection [17]. Synchronization among TCP connections that are transported within a common ATM virtual channel can give rise to the effects discussed above.

5.5 Simulation Results for a Client-Server Topology

We close this section with some simulation results from a final network configuration, one that is encountered frequently. The network environment we consider is a client-server configuration, a simple model of which is shown in Figure 5.14. In this example configuration, three remote nodes communicate with a central server. A pair of TCP connections, one in each direction, is set up between each remote node and the server node. Thus, there are three TCP connections originating at the server node, and three in the opposite direction. Each client-server connection is carried over a distinct ATM virtual channel, with the acks of a connection sharing the same VC used to transport data segments of the reverse connection. In this example, we set the IP service time to be infinitesimally small, forcing all the queueing of outgoing traffic to take place at the ATM layer. In the server, the ATM layer provides a separate queue per individual VC. The transmission rate of each VC, as computed by the rate allocation algorithm, is 51.6 Mbits/second. The maximum window size is 64 Kbytes (7 segments).

Figures 5.15 and 5.16 summarize the simulation results. Observe that even though traffic between different pairs of end-nodes is transported over different VCs and is therefore completely isolated, the effects of two-way traffic within each pair of connections is still present. The overall efficiency achieved is approximately 70%.

6 Conclusion

In this paper, we examined the performance of bidirectional TCP/IP connections over the Available Bit Rate class of service in ATM networks. We find that the problems of ack compression observed in packet networks manifests itself again in ATM networks, although the queueing of acks behind data packets is now primarily at the end systems. We observe the penalty of ack compression that has been found in the past: bunching of data packets flowing in either direction, and throughput degradation as a result of the periodic behavior of the queue going through idle intervals. The conclusion in the past was that the bunching occurring in the bottleneck links in the network was the primary reason for this behavior, and that a rate-controlled environment for transmitting data on the network would mitigate its effects. Our observation is that, although the queue lengths at the switches can be maintained small in an ATM network, the effect of ack compression persists, and may cause severe throughput degradation. For example, with symmetric bandwidths in each direction and window sizes large enough to fill the round-trip pipe, the throughput of each connection is only 66.67% of that under one-way traffic. The throughput degradation is even more dramatic when the transmission rates allocated to connections in opposite directions are widely different, even if the window of each connection is matched to the bandwidth-delay product in its direction. This is likely to be a common case with cable access or satellite links where the bandwidth in one direction can be large compared to that in the opposite direction. For example, with a one-way delay of 5 milliseconds, window size of 128 Kbytes, and transmission rates of 155 Mbits/second and 30 Mbits/second in the two directions, the high-speed connection is able to achieve only 20% of its throughput under one-way traffic.

Unlike earlier measurement- and simulation-based observations of ack compression, running TCP/IP over ATM allows us to analyze and predict the queue behavior at the sources of the bidirectional TCP connection (modeled as two uni-directional TCP connections in our analysis and simulations). Starting at the point where the TCP connection reaches a known steady state, we can arrive at closed-form expressions for both the end-system queue behavior as well as the throughput degradation with bidirectional traffic.

Our observation is that the effects of bidirectional TCP connections are prevalent in a wide range of situations, and it is therefore important to analyze and quantify them. We observed these effects are not only seen for two nodes in a simple configuration, but also in more complex configurations. For example, we observed the effect in a complex configuration of seven switches (a version of the GFC-2 configuration)[13] and multiple bidirectional TCP connections, along with some raw ATM cell cross-traffic as well. The effect is also present when there are several bidirectional TCP/IP conversations between a pair of end systems. The use of per-VC (virtual channel) queues at the end systems does not offer us a cure when the data and acks of a bidirectional TCP conversation share a common VC.

In an earlier paper [12] we examined the performance of TCP connections having very different round-trip times (RTT) in an ATM network. The maintenance of a current bandwidth allocation per virtual channel (VC) at the ATM switches was shown to achieve fairness in throughput superior to that in a datagram network. However, with bidirectional TCP traffic, the maintenance of a steady rate-allocation for each VC is no longer sufficient to achieve throughput fairness. The periodic behavior of the connections is now a function of the allocated rates, window sizes, and delays, causing significant disparity in throughput. Although the resources in the network are fairly allocated, and the flows may be virtually isolated from each other in the network, the efficiency of a connection with a smaller RTT in the bidirectional environment is better than that of a longer connection. This is primarily because of the excessive idle intervals imposed on the longer connection while it is awaiting acks.

The undesirable effects of two-way TCP traffic can be overcome by separating the TCP segments and acks of the same bidirectional TCP connection throughout the protocol stack in the end system and transporting them over separate ATM virtual channels, but this is wasteful of valuable resources. Alternatively, queueing outgoing acks at a higher priority eliminates ack compression, but requires support for message priorities in the protocol stack. This idea of processing acks at a higher priority was suggested by Mogul [4] in the context of IP routers. A subject of our future work will be to investigate these and other solutions to improve the performance of two-way TCP over ATM.

Bidirectional TCP conversations are not unusual, since most TCP connections are set up in this manner: the behavior we observe is expected to occur when both ends have data to send. Therefore, we believe that the general problem deserves further research, and that the insight and results presented here will be valuable in devising solutions to the problem.

References

[1] V. Jacobson, "Congestion avoidance and control," in *Proc. of ACM SIGCOMM'88*, pp. 314–329, 1988.

[2] V. Jacobson, "Modified TCP congestion avoidance algorithm." message to end2end-interest mailing list, April 1990.

[3] S. Floyd and V. Jacobson, "On traffic phase effects in packet-switched gateways," *Internetworking: Research and Experience*, vol. 3, no. 3, pp. 115–156, September 1992.

[4] J. C. Mogul, "Observing TCP dynamics in real networks," in *Proc. of ACM SIGCOMM'92*, pp. 305–317, August 1992.

[5] L. Zhang and D. D. Clark, "Oscillating behavior of network traffic: A case study simulation," *Intenetworking: Research and Experience*, vol. 1, no. 2, pp. 101–112, December 1990.

[6] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic," in *Proc. of ACM SIGCOMM'91*, pp. 133–147, September 1991.

[7] R. Wilder, K. K. Ramakrishnan, and A. Mankin, "Dynamics of congestion control and avoidance of two-way traffic in an OSI testbed," *ACM Computer Communication Review*, vol. 21, no. 2, pp. 43–58, April 1991.

[8] F. Bonomi and K. W. Fendick, "The rate-based flow Control framework for the Available Bit Rate ATM service," *IEEE Network*, vol. 9, no. 2, pp. 25–39, March/April 1995.

[9] D. E. Comer and J. C. Lin, "TCP buffering and performance over an ATM network," *Internetworking: Research and Experience*, vol. 6, pp. 1–13, 1995.

[10] S. Floyd and A. Romanow, "Dynamics of TCP traffic over ATM networks," in *Proc. of ACM SIGCOMM'94*, pp. 79–88, September 1994.

[11] L. Kalampoukas and A. Varma, "Performance of TCP over multi-hop ATM networks: A comparative study of ATM layer congestion control schemes," in *Proc. of ICC'95*, pp. 1472–1477, June 1995.

[12] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "Dynamics of an explicit rate allocation algorithm for ATM networks," in *Proc. of International Broadband Communications Conference'96*, IFIP-IEEE, April 1996.

[13] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "Two-way TCP traffic over ATM: Effects and analysis," Tech. Rep. UCSC-CRL-96-23, Univ. of California, Santa Cruz, 1996.

[14] W. R. Stevens and G. R. Wright, *TCP/IP Illustrated*, vol. 2. Addison-Wesley Publishing Company, 1995.

[15] S. S. Sathaye, *Traffic management specification, version 4.0.* Traffic Management Working Group, April 1996.

[16] L. Kalampoukas and A. Varma, "Analysis of source policy in rate-controlled ATM networks," in *Proc. of ICC'96*, IEEE, June 1996.

[17] R. Cole, D. Shur, and C. Villamizar, "IP over ATM: A framework document," *Request for Comments (RFC): 1932*, April 1996.