PERFORMANCE OF TCP/IP OVER ATM NETWORKS

MAHBUB HASSAN

MOHAMMED ATIQUZZAMAN

Performance of TCP/IP Over ATM Networks

For a complete listing of the *Artech House Telecommunications Library,*
turn to the back of this book.

Performance of TCP/IP Over ATM Networks

Mahbub Hassan
Mohammed Atiquzzaman

Artech House
Boston • London
www.artechhouse.com

Library of Congress Cataloging-in-Publication Data
Hassan, Mahbub.
Performance of TCP/IP over ATM networks / Mahbub Hassan, Mohammed Atiquzzaman.
p. cm.—(Artech House telecommunications library)
Includes bibliographical references and index.
ISBN 1-58053-037-0 (alk. paper)
1. Asynchronous transfer mode. 2. Telecommunication—Traffic.
I. Atiquzzaman, Mohammed. II. Title. III. Series.
TK5105.35.H37 2000
004.6'2—dc21 00-040622
CIP

British Library Cataloguing in Publication Data
Hassan, Mahbub
Performance of TCP/IP over ATM networks. — (Artech House telecommunications library)
1. TCP/IP (Computer network protocol) 2. Asynchronous transfer mode
I. Title II. Atiquzzaman, Mohammed
004.6'2
ISBN 1-58053-405-8

Cover design by Gary Ragaglia

685 Canton Street
Norwood, MA 02062

International Standard Book Number: 1-58053-037-0
Library of Congress Catalog Card Number: 00-040622

10 9 8 7 6 5 4 3 2 1

To my parents for their love and sacrifice
and to my son Aaron and my wife Jahan
Mahbub Hassan

To my parents for their support and constant encouragement,
and to Nafis, Samia, and Mobina for putting up with the long
hours I had to spend while writing this book
Mohammed Atiquzzaman

Contents

Foreword

ATM introduced four key features to the networking world—traffic management, Quality of Service (QoS), signaling, and service integration. The sophisticated traffic management techniques used by ATM were specifically designed for high-speed networks. ATM has feedback and negotiation mechanisms that allow traffic sources to be throttled without loss. All other competing technologies have either no traffic management or use loss-based mechanisms to handle traffic. In very high-speed networks, loss-based techniques can lead to long delays and huge losses. ATM also has several service classes specifically designed to meet various QoS requirements. Delay-sensitive and delay-insensitive traffic are queued and serviced separately. QoS requires mechanisms for users to specify their traffic patterns and QoS requirements for the network. This is known as signaling. And finally, ATM was designed to allow the integration of data, voice, and video so that telecommunications providers (voice networks) could move easily into the data market.

ATM design was, however, dominated by the telecommunications carriers and equipment vendors. As a result, some design decisions—although

ideal for voice—have not been optimal for data. The selection of ATM cell size is a prime example. Today, ATM is used extensively throughout the carrier networks. Most of the big telecommunications carriers have switched their core network to ATM. By some estimates, over 80% of Internet traffic already passes through ATM networks. However, ATM has failed to reach the desktop. Enterprise networks are still primarily data-oriented, with voice and video coming in slower than projected. This has given competing technologies enough time to develop their own traffic management, QoS, signaling, and service integration mechanisms. Today we are all busy developing these mechanisms for TCP/IP and Ethernet. The mechanisms developed for ATM have an important impact on these developments because the research and knowledge developed for ATM can be used, enhanced, and optimized for these other data-oriented technologies.

The reality today is that TCP/IP predominates in the enterprise networks while ATM occupies the core telecommunications networks. This scenario may change tomorrow, but the performance of TCP/IP traffic over ATM is key to getting the most out of the networks as they exist today.

A number of books have been published describing the techniques used to interconnect TCP/IP and ATM. Hassan and Atiquzzaman's book is unique in that it deals with the performance issues involved in interconnecting TCP/IP and ATM. The book starts with brief descriptions of TCP/IP and ATM, outlines schemes for transporting TCP/IP over ATM, and then goes into an in-depth discussion of the issues affecting the performance of TCP/IP over ATM networks. The need for mobility has given rise to a strong interest in being able to access the Internet over wireless satellite links. The last chapter of this book discusses enhancements to enable TCP over long delay satellite links. The material presented in this book is very up-to-date and relevant to researchers and developers of TCP/IP over ATM equipment for next-generation data networks.

This book is an excellent reference for seniors and graduate students. It can also be beneficial for practitioners and scientists in the field of networking. This will serve as a definitive text on TCP/IP over ATM networks.

Raj Jain
Professor of Computer and Information Science
Ohio State University

Preface

Transmission Control Protocol (TCP) is the transport protocol that supports the most popular Internet applications, such as the World Wide Web, file transfer, and e-mail. The performance of these applications significantly depends upon the performance of TCP. Low TCP throughput is one of the main reasons behind the long download times experienced with many Internet connections. It should be mentioned that TCP was developed for non-real-time data applications and hence does not provide any Quality of Service (QoS) guarantees for real-time applications.

Asynchronous Transfer Mode (ATM) has been developed for providing QoS guarantees for real-time applications. However, the high cost of ATM equipment, the large installed base of TCP/IP-based applications, and the lack of application programs utilizing pure ATM switched connections has resulted in ATM being used as a lower layer technology at the core of Wide Area Networks, with TCP/IP at the edge. This is evidenced by the increasing deployment of ATM networks in the building of high-speed Internet and intranet backbones.

The rapid deployment of ATM backbones in TCP/IP networks has sparked a flood of research on the performance dynamics of TCP over ATM networks. However, the results of this research remain scattered in conference proceedings, journals, and standards documents.

This book is an attempt to explain the issues and collate the published results for TCP performance over ATM networks in a single reference. This book will therefore be valuable to anyone who needs to understand the performance issues and answers for running TCP-based applications over ATM networks. Faculty and graduate students doing research on TCP performance will particularly benefit from this book, as will network administrators and Internet service provider (ISP) operators deploying ATM in the backbones of existing TCP/IP networks.

Although some general background in computer networks is necessary to fully comprehend the materials in this book, domain knowledge in TCP and ATM is not a prerequisite. Introductory chapters are included in the book for readers without prior knowledge of TCP and ATM.

A Web site for this book will be maintained at http://www.tcpperformance.com/tcpatm/ to provide additional support for readers. This site will contain an updated errata list, class notes and presentation materials based on the book, and links to relevant sites.

We would like to thank several people at Artech House for their help. Mark Walsh, the Senior Acquisitions Editor, is the person responsible for provoking the first author (Mahbub Hassan) into writing this book in the first place. We thank Assistant Editor Barbara Lovenvirth for the role she played in getting our chapters reviewed, adjusting missed deadlines, and working as a liaison between Artech House and ourselves. We also thank Sean Flannagan (Production Editor), Judi Stone (Executive Editor), Jack Stone (Director of Marketing), and others from Artech House for their work in the production and marketing of this book. We are grateful for the many useful comments made by the anonymous technical reviewer appointed by Artech House as well. And, finally, we extend our gratitude to Professor Raj Jain for writing the foreword.

Mahbub Hassan
Mohammed Atiquzzaman

1 Introduction

1.1 Transmission control protocol/Internet protocol (TCP/IP)

The Internet has become an integral part of life in much of the world. In fact, there are too many uses of the Internet to describe. Common uses of the Internet include sending electronic mail (e-mail), investing in the stock market, making phone calls, watching television, and advertising on the World Wide Web (WWW). The Internet revolution is the most significant of the century. As evidence of the phenomenal growth of the Internet, consider the growing number of Internet hosts. See Figure 1.1.

The success of the Internet in the public domain is now penetrating corporate boundaries. Many enterprises are adopting Internet technology for their internal corporate networks, or *intranets*. Intranets use the same applications and technology as the Internet—including e-mail and the WWW—but are protected from the public Internet using appropriate security.

The centerpiece of the Internet/intranet technology is the transmission control protocol/Internet protocol (TCP/IP) suite. IP is the

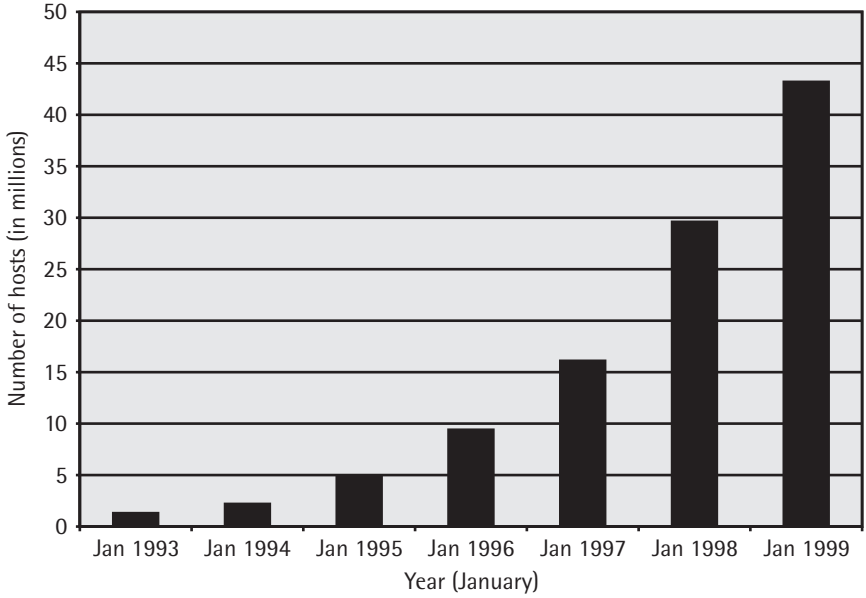


Figure 1.1 Number of Internet hosts. (*Source:* Internet Software Consortium: http://www.isc.org.)

internetworking layer that glues together millions of hosts worldwide; it is the global interoperability solution for diverse hardware and networking platforms on the Internet. The IP layer hides the details of the underlying physical networks and provides a common platform for all Internet hosts, as depicted in Figure 1.2.

TCP is a transport layer protocol on top of IP that provides reliable, connection-oriented services to the application layer. TCP uses retransmission to recover from any packet loss in the network. User datagram protocol (UDP) is a lightweight counterpart of TCP that does not implement complex functions such as retransmission and does not guarantee delivery of packets. Most Internet applications, including the popular HTTP for WWW, use TCP. However, due to protocol complexity and processing delay, some application protocols—such as the Simple Network Management Protocol (SNMP)—use UDP.

Figure 1.3, which illustrates the layered architecture of TCP/IP protocol suite, shows that most popular Internet applications use the reliable TCP protocol. Internet traffic studies [1] show that most of the traffic

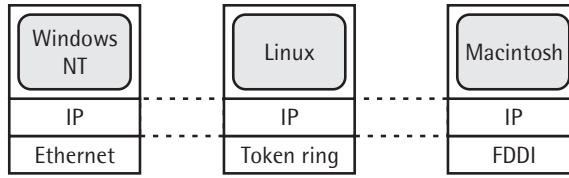


Figure 1.2 Three hosts with different software and hardware are connected using the common IP platform.

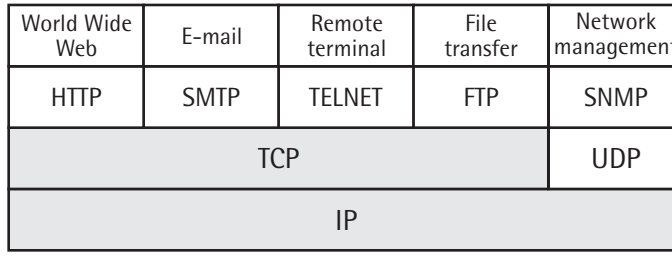


Figure 1.3 TCP/IP suite.

comes from TCP, with UDP generating only a small fraction of the total traffic.

1.2 TCP/IP networking with ATM networks

The International Telecommunications Union (ITU) has recently standardized Asynchronous Transfer Mode (ATM) as the transmission technology for the Broadband Integrated Services Digital Network (B-ISDN). ATM, a high-speed transmission technology capable of operating at hundreds of megabits per second, has been designed to integrate voice, video, and data on the same network.

Due to unprecedented traffic growth on the Internet, traditional transmission technologies, such as 1.544-Mbps T1 links, are stretching to cope with the ever-increasing volume of TCP/IP traffic. ATM's high speed makes it a viable solution to the bandwidth problem of today's TCP/IP networks. Accordingly, large Internet service providers (ISPs), such as MCI, are adopting ATM as their high-speed backbone network. ATM is also

being used by large enterprises to replace leased-line–based local area network (LAN) interconnections between long-distance offices. Although the adoption rate was slower than expected in the beginning due to the high cost of ATM equipment, ATM is now gaining momentum, especially as a solution for high-speed data networks. Hundreds of thousands of ATM switches have been sold worldwide in the last few years. Although there are some alternative high-speed solutions, such as fast and gigabit Ethernet in the local area and Synchronous Optical Network (SONET) and Wave Division Multiplex (WDM) in the wide area, it is expected that in the near future, a large volume of TCP/IP traffic will be transported over ATM networks.

TCP/IP and ATM are based on completely different protocols. TCP/IP is based on a connectionless best-effort service, whereas ATM is a connection-oriented service that provides quality of service (QoS) guarantees to applications. To provide QoS guarantees, ATM uses small packet sizes; TCP/IP uses much larger packet sizes.

A number of issues result from the interconnection of two different types of networks. For example, large TCP/IP packets need to be broken down into smaller ATM packets at the point of transition from TCP/IP to ATM networks, and new headers have to be added to every ATM packet to enable routing through the ATM network. A second issue would be setting up connections in the ATM network at the point where TCP/IP packets enter the ATM network. Setting up connections in an ATM network gives rise to delays, and the TCP/IP packets have to be buffered at the gateway between the two networks. Since the two networks use different congestion-control techniques, a third issue would be the interaction between the two congestion-control techniques. This book aims to quantitatively study the performance of running TCP/IP over ATM networks in the presence of such internetworking issues and to solve problems that arise.

1.3 Significance of TCP/IP performance

The most prominent Internet applications, such as WWW and e-mail, are based on TCP/IP. The implementation of these applications and, in turn, the networking performance of TCP/IP-based enterprises, therefore, critically depend on how TCP/IP performs. Moreover, as the number of

TCP-based applications continues to grow, TCP performance becomes more critical than ever. Figure 1.4 shows the explosive growth of the WWW.

Now that we appreciate the importance of TCP/IP performance in general, we turn to TCP/IP performance over ATM networks, the central theme of this book. Since its market release in 1995 by large telecommunications carriers, public ATM service continues to enjoy growth in its customer base. Today, many carriers, such as AT&T, Sprint, US West, and Ameritech, have started offering ATM services to their customers. Figure 1.5 shows ATM's customer base growth for 1998–1999 in different market segments.

Although ATM has not become the world's "one and only" network, it has certainly become competitive in the LAN backbone and wide area network (WAN) market. At present, in fact, ATM enjoys about 11% of the LAN backbone market, and it is predicted to achieve 30% [2] in only five years. Moreover, according to a Yankee Group market forecast [2] for 2001, ATM will be the most predominant Metropolitan Area Network (MAN) and WAN technology; it will account for almost 75% of the total market share, as shown in Table 1.1.

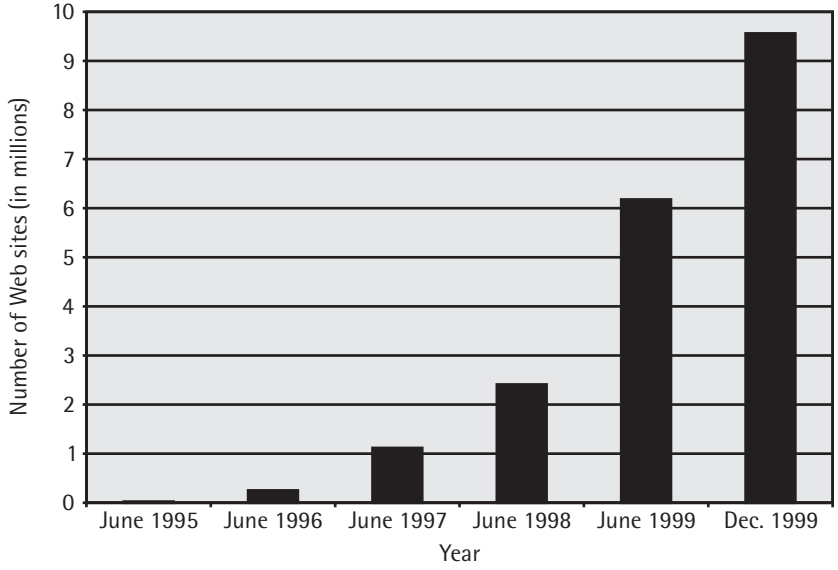


Figure 1.4 Growth of the WWW. (*Source:* Hobbes' Internet Timeline: http://www.isoc.org/zakon/Internet/History/HIT.html. ©1993–2000 Robert H Zakon.)

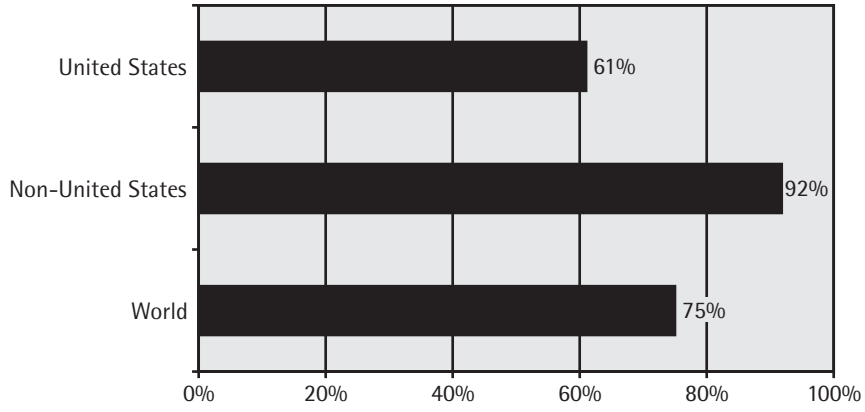


Figure 1.5 Customer base growth for public ATM services during the period 1998–1999. (*Source:* http://www.webtorials.com.)

Table 1.1
Projected Market Share of ATM as MAN and WAN Technology in 2001

Network Technology	Revenue (millions of U.S. dollars)	Backbone Platforms (percentage)
Fiber Distributed Data Interface (FDDI) and private lease line	98	11
Packet over SONET	166	15
ATM	842	74

Source: [2].

The growing acceptance of ATM in the LAN and WAN backbones suggests that a large volume of TCP/IP traffic is and will be carried over ATM networks. It is, therefore, important to study the performance issues for TCP/IP over ATM networks. Network engineers and operators using ATM technology to transport TCP/IP traffic must address these performance issues to build high-performance TCP/IP networks. Insufficient understanding of such issues and a failure to properly configure the networks will adversely affect the performance of intranet applications of TCP/IP over ATM. Poor performance issues include low throughput, poor response time, high packet loss, and high variation in response times.

These issues will, in turn, affect the performance of the applications running over TCP/IP.

1.4 TCP/IP and ATM standards bodies

To promote interoperability among different vendor products, it is essential to standardize techniques and products for TCP/IP networking over ATM networks. Three standards bodies—the Internet Engineering Task Force (IETF), the ITU, and the ATM Forum—have been developing standards for various aspects of TCP/IP over ATM. Table 1.2 lists these groups' official Web sites, where one can find information on many standardization activities. Sections 1.4.1–1.4.3 briefly describe each of these bodies.

1.4.1 IETF

The IETF has been the standards organization for the Internet, so naturally when it became necessary to transport TCP/IP over ATM networks, IETF got involved in the standardization process. Standards documents by IETF, which are published as Requests for Comments (RFCs), are freely distributed through several public file transfer protocol (FTP) servers around the world and through the official IETF Web site (see Table 1.2). Initial submissions to IETF, or Internet drafts, are valid for six months; after six months, an Internet draft expires, and it is removed from the FTP server. Some important Internet drafts are promoted to RFC status.

Relevant RFCs include RFC 793 [3] for TCP, RFC 791 [4] for IP, and RFC 2225 [5] for IP over ATM. The IETF's official Web site offers a complete list of all the RFCs related to TCP/IP over ATM.

Table 1.2
Web Sites for Relevant Standardization Bodies

Standardization Body	Web Site
IETF	http://www.ietf.org
ITU	http://www.itu.int
ATM Forum	http://www.atmforum.com

1.4.2 ITU

In 1988, the ITU originally proposed ATM as the base networking technology for B-ISDN. Since then, the ITU has published many standards on ATM, known as I-series recommendations. However, the ITU has mainly focused on ATM technology, with little discussion of transporting TCP/IP over ATM. Unlike IETF information, ITU documents are not freely distributed, but they are available for a fee.

1.4.3 ATM Forum

Because of the long delays involved in the standardization process of ATM in the ITU, a group of computer and communication industries jointly formed a separate body, known as the ATM Forum, to speed the standardization process in 1991. The ATM Forum was primarily driven by the computer and data networking industries, which saw ATM as a promising technology for high-speed data communications. The ATM Forum is now a truly international forum whose members span the industry, government, and education sectors and many countries worldwide.

Submissions to the ATM Forum by member organizations are called ATM Forum contributions. Since the driving force behind the ATM Forum was data communications, there have been numerous contributions to the ATM Forum on TCP/IP over ATM. ATM Forum contributions are made available free of charge to ATM Forum members only.

ATM Forum contributions are working (or interim) documents (similar to IETF drafts) in the process of standardization. Once a document is standardized, it is called an ATM Forum specification. These specifications are publicly available and can be downloaded free of charge from ftp://ftp.atmforum.com/pub/approved-specs/. There are hundreds of specifications addressing various issues ranging from traffic management and signaling, to user interface. Traffic Management Version 4.0 [6] is an example of an ATM Forum specification; it outlines, among other things, various rules for accessing the available bit rate (ABR) service of ATM networks.

1.5 Organization of the book

The rest of this book is organized into ten chapters. Chapter 2 familiarizes readers with the fundamentals of TCP and IP protocols; readers need this

background to understand the performance issues discussed in subsequent chapters. Readers with solid knowledge of TCP/IP may skip Chapter 2.

To understand the performance issues for TCP/IP over ATM networks, one must be familiar with the basics of ATM networks. Chapter 3 serves this purpose. Readers already familiar with ATM networks may skip Chapter 3.

Chapter 4 describes the architectures used to transport TCP/IP traffic over ATM networks. Readers need to be familiar with the materials in Chapters 2 and 3 to follow Chapter 4. Next, Chapter 5 provides an overview of various performance issues for transporting TCP/IP traffic over ATM networks—namely ATM protocol overhead, ATM cell loss, ATM virtual circuit (VC) setup delay, interactions between ATM rate control and TCP congestion control algorithms, and large maximum transfer unit (MTU) size for IP over ATM.

Chapters 6–10 discuss techniques and solutions addressing each of the performance issues introduced in Chapter 5. When multiple solutions are possible to address a performance issue, their merits, limitations and performance are compared. Such comparisons will help network managers to select the right alternatives for their operating environments.

Due to their global coverage and broadcast capabilities, satellite ATM networks have recently attracted the attention of network service providers to provide high-speed data services on a global scale. Since satellite networks suffer from longer delays and higher error rates than those for which the TCP was initially designed, Chapter 11 discusses the various TCP enhancements for operation over satellite links and the performance dynamics of TCP over satellite networks.

References

[1] Thompson, K., G. J. Miller, and R. Wilder, "Wide Area Internet Traffic Patters and Characteristics," *IEEE Network,* Vol. 11, No. 6, November/December 1997, pp. 10–23.

[2] Black, U., *ATM Volume I Foundation for Broadband Networks,* Second Edition, Englewood Cliffs, NJ: Prentice Hall, 1999.

[3] Postel, J., "Transmission Control Protocol," *Internet RFC 793,* IETF, September 1981.

[4] Postel, J., "Internet Protocol," *Internet RFC 791,* IETF, September 1981.

[5] Laubach, M. and J. Halpern, "Classical IP and ARP over ATM," *Internet RFC 2225,* IETF, April 1998.

[6] ATM Forum, "Traffic Management Specification v4.0," 1996.

2

TCP and IP

2.1 Introduction

Most TCP/IP over ATM performance issues arise from various interactions between TCP/IP and ATM protocols. To understand these performance issues and the techniques addressing them, one has to be familiar with some of the basic features of TCP/IP protocols. Several books [1, 2] provide in-depth coverage of the protocols, architectures, and implementations of TCP/IP. This chapter reviews only those features of TCP/IP relevant to understanding the subject matters in the later chapters.

2.2 TCP

This section explains the following aspects of TCP:

- Maximum segment size (MSS);
- Segment header format;
- Retransmission mechanism;

- Flow control;
- Congestion control.

2.2.1 MSS

TCP transmits application data in units of bytes called segments. The MSS for a TCP connection determines the maximum number of application bytes that can be carried in one TCP segment, which, in turn, is encapsulated in an IP datagram.

Since there is some overhead at TCP and IP and in the lower layers for every TCP segment, it is desirable to select an MSS as large as possible to minimize protocol overhead per application byte transmitted. TCP, therefore, usually selects an MSS that, after adding TCP and IP header bytes, will just fit in the frame payload of the underlying network. The maximum payload of the underlying network is the MTU. For example, Ethernet has an MTU of 1,500 bytes. Accordingly, for 20 bytes of TCP and 20 bytes of IP header, TCP would select an MSS of 1,500 − 40 = 1,460 bytes for communication over Ethernet.

2.2.2 Header format

The unit of data transfer between two TCP end points is a segment. A TCP segment has two parts: a header followed by a payload. The header carries necessary control and identification information, and the payload carries application data. Figure 2.1 shows the TCP header format.

There are many fields in the TCP header, each serving a different purpose. To understand TCP protocol operations, it is necessary to examine

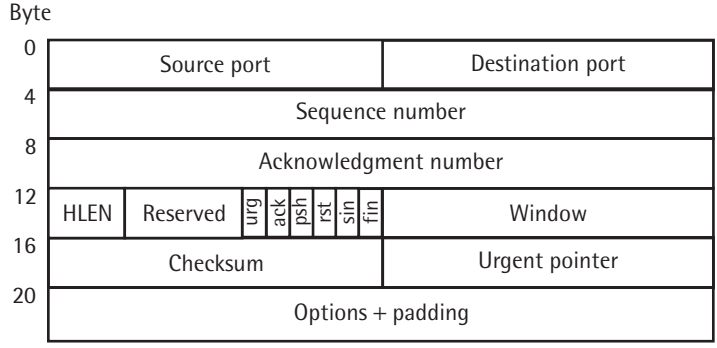


Figure 2.1 TCP header format.

the meaning and purpose of these fields. TCP header fields are described in the following sections.

Source port (16 bits)

The 16-bit SOURCE PORT field contains the TCP port number identifying the application at the source host. The use of port numbers to identify a given application allows multiplexing of several applications at the end hosts onto the same TCP software.

Destination port (16 bits)

The 16-bit DESTINATION PORT field contains the TCP port number identifying the applications at the destination host.

Sequence number (32 bits)

For segments carrying data, the 32-bit SEQUENCE NUMBER field identifies the first byte in the payload in the byte stream from the application. For example, if the sequence number is 500 and the segment carries 1,000 bytes of data in the payload, the sequence number of the next segment will be 1501. It is important to emphasize that TCP identifies a segment using this byte-based sequence number; it does not use a separate sequence number for the segments.

If the segment is a SYN segment, or the first segment of a TCP connection trying to establish a connection, there is no payload. For a SYN segment, the sequence number field carries the Initial Sequence Number (ISN) of the connection. The first data byte in the first data segment will be numbered ISN + 1.

Acknowledgment number (32 bits)

The 32-bit ACKNOWLEDGEMENT NUMBER field is used by the destination to identify the first byte it expects to receive. In effect, the destination acknowledges all bytes before the one identified by the number in this field. Two observations are in order.

1. The above acknowledgment process allows the destination to acknowledge more than one segment in one acknowledgment.
2. The acknowledgment of a segment is done through a field in the TCP header that allows an acknowledgment to piggyback a data segment from destination to source.

HLEN (4 bits)

The minimum and the typical TCP headers are 20 bytes. However, if the OPTIONS field is used, the length of the TCP header varies depending on which options are used. The length of the TCP header in 32-bit words is contained in the 4-bit HLEN field. The TCP header is an integral number of 32-bit words.

Reserved (6 bits)

The 6-bit field next to HLEN is reserved for future use.

Flags (6 bits)

A TCP segment may carry, for example, normal data, some urgent data, and an acknowledgment, among other things. Clearly, some mechanism is necessary to determine what type of data or message is carried by the segment. Each bit of the 6-bit flags field is used as a flag to indicate the type of message carried by the segment. Table 2.1 shows the meaning of each bit when it is set.

Window (16 bits)

As will be explained in Section 2.2.4, TCP implements a window-based flow control mechanism to prevent a slow receiver from being swamped by a fast sender. The receiver updates the sender's window size depending on

Table 2.1
Meaning of the Bits in the Flags Field of the TCP Header

Bits in Flags Field (left-to-right)	Meaning
URG	Urgent pointer field is valid
ACK	Acknowledgment field is valid (carrying acknowledgment)
PSH	PUSH function: Ordinarily, TCP accumulates data until there is enough data to form a "large" segment; if the PSH bit is set, it tells TCP to send all pending data without waiting for more data to accumulate
RST	Reset the connection
SYN	A new transfer starts; synchronize the sequence numbers
FIN	A transfer completes; this is the last segment from sender

the available buffer to control the flow of data from sender to receiver. The 16-bit WINDOW field is used for updating the window.

Checksum (16 bits)

The 16-bit CHECKSUM field is used to check the integrity of both header and payload.

Urgent pointer (16 bits)

TCP segments may carry urgent data that requires special attention at the receiver. The urgent pointer field points to the last byte of urgent data carried in a segment. This allows the destination to determine the end of urgent data.

Options (variable)

All options are multiples of eight bits in length. One of the formats specified in RFC 793 [3] stipulates that each option should be preceded by two bytes, the first byte identifying the option type and the second byte containing the length of the option in bytes including these two preceding bytes. Three important options have been defined so far:

- MSS (16 bits): This option was defined in the original RFC 793 [3].This 16-bit option is used only in SYN segments to allow the originator of the TCP connection to negotiate the MSS in bytes. With 16 bits, the maximum segment size that can be negotiated is 2^{16}, or 64, KB.
- Window scale factor (8 bits): This option was defined only a few years ago in RFC 1323 [4] to eliminate TCP window size bottleneck for large delay bandwidth links (long fat pipes).Usually the window field in the TCP header contains the size of the window expressed in bytes. For a 16-bit window field, this allows a maximum of a 64 KB window, which may not be sufficient to fill the pipe for very high-speed links. The window scale factor option allows the TCP connection originator to negotiate a much larger window in the SYN segments. When this option is used, the number in the window field is multiplied by 2^F, where F is the value of the window scale option. The maximum value of F accepted by TCP is 14. Therefore, the maximum window size possible is $2^{16} \times 2^{14} = 2^{30}$, or 1 GB.

- Timestamp (8 bytes): This option carries two 4-byte timestamp fields as defined in RFC 1323 [4]. The first 4-byte field is used by sending TCP to timestamp a TCP segment. The receiver echoes the timestamp value sent by the sender in the second 4-byte field in an ACK segment. Unlike the previous two options, this option can be used in any data segment to include the current time. This option allows TCP connections to monitor round-trip times more accurately.

Padding (variable)

Padding of all zero bytes is used to ensure that the TCP header is an integral number of 32-bit words.

2.2.3 Retransmission

TCP handles the packet loss problem in unreliable networks by retransmitting the lost packets. Every time TCP sends a segment, it starts a timer and waits for acknowledgment from the receiver. If the timer expires before the acknowledgment arrives, TCP assumes that the segment is either lost or corrupted and retransmits the entire segment.

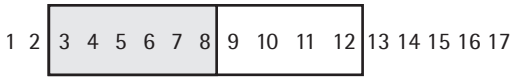
2.2.4 Flow control

TCP uses byte-based window flow control to restrict the number of bytes in transit (transmitted but not yet acknowledged). The size of the window controls the number of bytes in transit; the larger the window, the more bytes can be transmitted before waiting for acknowledgment.

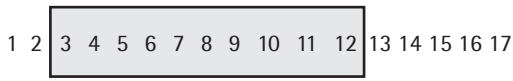
When a window full of data is in transit, TCP must stop transmitting and wait for acknowledgment. When an acknowledgment arrives, TCP can transmit new data not exceeding the number of bytes just acknowledged.

Figure 2.2 illustrates the sequence of bytes that are inside and outside the window of size 10 at three different times. At time t_1, bytes 1 and 2 have been acknowledged; TCP has transmitted bytes 3–8 and may transmit bytes 9–12 without waiting for acknowledgment. Transmission of bytes 9–12 is completed at time t_2, and TCP starts waiting for acknowledgment. Bytes 3–6 are acknowledged at time t_3, and the window is advanced by 4 bytes to allow the TCP to transmit another 4 bytes.

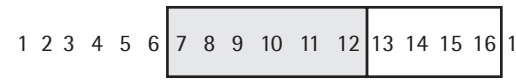
TCP implements *dynamic* window flow control where the window size can be varied during a given TCP session. The sender's window size may be



Time t_1: TCP has transmitted bytes 3–8 and is allowed to transmit 9–12 without waiting for acknowledgment



Time t_2: TCP has transmitted bytes 3–12 and is forced to wait until it receives acknowledgments



Time t_3: TCP has received acknowledgments for bytes 3–6 and is allowed to transmit bytes 13–16

Figure 2.2 Window flow control of TCP.

dynamically updated by the receiver via the window advertisement fields in acknowledgment packets. This way, a slow receiver (e.g., a PC) can prevent a fast transmitter (e.g., a supercomputer) from swamping its buffers.

2.2.5 Congestion control

In addition to the overflow problem between a fast sender and a slow receiver, TCP faces another big problem—network congestion. When an IP router in the network starts to receive datagrams at a rate higher than it can route, the router begins to queue the incoming datagrams in the buffer (temporary storage). If the queue becomes long and continues to grow, the situation is called congestion.

Routers have finite storage capacity. If congestion persists, buffers become full, and eventually some incoming datagrams are lost in the congested router. To control network congestion, TCP must slow down until the congestion is alleviated.

Unfortunately, window flow control through dynamic window update by the receiver is not effective in controlling network congestion. The

receiver is not aware of the congestion in the network and hence cannot reduce the window size during congestion.

As a result, TCP will time-out and retransmit. The retransmissions will cause more congestion, leading to increased delay and packet loss, which will cause further retransmissions and so on, until the network collapses. This phenomenon is known as congestion collapse.

To prevent congestion collapse, TCP implements three additional mechanisms—multiplicative decrease, slow start, and congestion avoidance. With multiplicative decrease, TCP reduces its window by half each time a timer expires down to a minimum of one segment. This reduces traffic exponentially, quickly and effectively responding to network congestion.

When congestion is over (i.e., TCP starts to receive acknowledgments, or TCP starts a new connection and is not aware of the congestion status of the network), TCP implements slow start. With slow start, TCP starts with a window size of one segment and increases the window by one each time a segment is acknowledged. The idea is to increase the traffic slowly.

Interestingly, slow start is not actually that slow; it increases the window exponentially as a function of round-trip times (RTTs). To understand the exponential increase, it is necessary to examine the window increase process from the start. Initially, the window size is one segment. After the first acknowledgment arrives in one RTT, the window is increased to two, and TCP transmits two segments at the same time. In another RTT, these two segments are acknowledged, leading to a window of four segments. If the process continues, in the next RTT, the window will be eight and then sixteen in the following RTT and so on, leading to an exponential increase in window size.

If unchecked, exponential traffic increase during the slow start period will quickly lead to another congestion collapse. To avoid such periodic congestion collapse, TCP implements a clever algorithm called congestion avoidance to enforce a linear window increase after the window reaches half of the original size when congestion first occurred. During congestion avoidance, TCP increases the window by one every RTT.

Figure 2.3 shows the growth of the TCP window as a function of RTT during slow start and congestion avoidance, assuming that the congestion first occurred (retransmit timer expired) when the window was 32 segments in size. Figure 2.3 shows that the window increases exponentially until it reaches 16 (half of 32); after that it enters congestion avoidance and grows linearly.

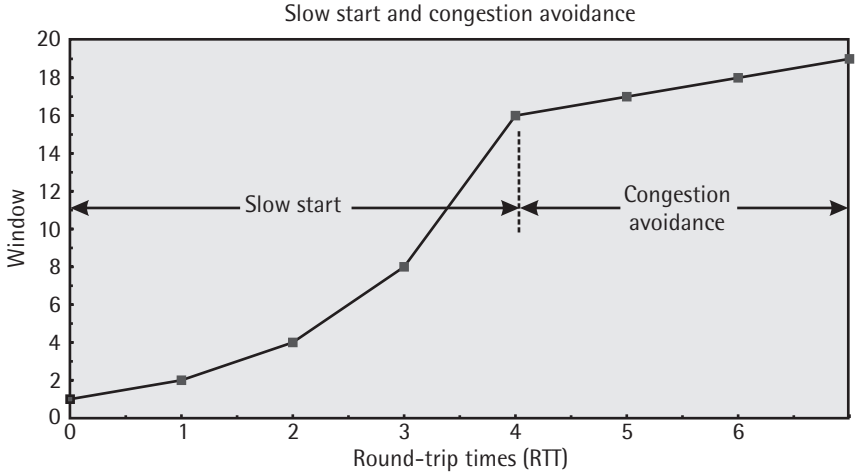


Figure 2.3 Window increase during slow start and congestion avoidance.

2.3 IP

IP is the common network layer protocol used by all protocols in the TCP/IP protocol suite. This section provides an overview of selected aspects of the IP protocol. For a comprehensive description of IP, readers may consult [1, 2].

2.3.1 Unreliable delivery

IP provides an unreliable service to the software or application that uses it. The service is called unreliable because IP makes no attempt to make sure that a datagram is correctly delivered to the destination, although it tries its best to do so. As a result, the datagram may be dropped or duplicated or arrive out of order at the destination.

2.3.2 Connectionless delivery

IP is a connectionless protocol. There is no connection setup or teardown between source and destination IP modules. Each IP datagram is treated independently as a separate message of its own with explicit destination information included in the header. Connectionless delivery allows routers to choose different routes for different datagrams between the same source and destination hosts depending on the congestion or route availability.

Unfortunately, if different datagrams travel through different routes, out-of-order arrivals are possible as the delays experienced could be different in different routes.

2.3.3 IP Header format

The unit of data transfer between two IP entities is a datagram. Like a TCP segment, an IP datagram has two parts, a header followed by a payload. The header carries necessary control and identification information, and the payload carries data from TCP, UDP, or any other higher layer protocol running on top of IP.

The IP protocol used in the Internet is called IP version 4, or simply IPv4. The IETF has recently standardized a new version called version 6 or IPv6 to be used in the next-generation Internet. IPv6 has a different header format than IPv4. This section presents the header format for IPv4. Figure 2.4 shows the IPv4 datagram header format.

Like the TCP segment header, the minimum and typical IP header is 20 bytes long, and it is sometimes augmented with some optional fields. The fields in the IP header are described in the following sections.

Protocol version (4 bits)

The version number of the IP protocol is placed in this field. For the current Internet, this field contains the number 4. The next-generation IP protocol has the version number 6.

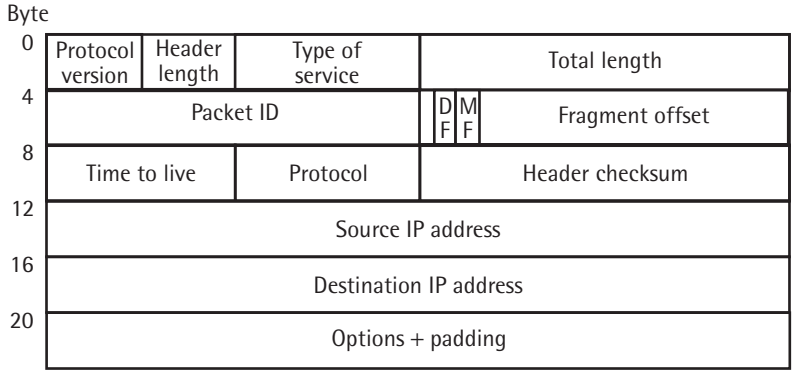


Figure 2.4 IPv4 datagram header format.

Header length (4 bits)

The header length indicates the length of the IP header in multiples of 32-bit words. This field is necessary, because the IP header has a variable size when options are used.

Type of service (8 bits)

Using this field, the source IP module can request the intermediate routers along the path of destination to provide special treatment to the datagram. These special requests include minimizing delay and maximizing throughput. However, intermediate routers are not obliged to honor any such request. In the current Internet, this field is not used.

Total length (16 bits)

The total length indicates the total datagram length in bytes. For a 16-bit field, the maximum datagram size allowed is $2^{16} - 1 = 65{,}535$ bytes, including the header bytes. A total length field is necessary to detect the end of the datagram, as IP datagrams do not have an end-of-datagram field.

Packet ID (16 bits)

This field uniquely identifies a datagram with a sequence number. The number is incremented by one each time a new datagram is sent.

Flags (3 bits)

IP datagrams may be fragmented by intermediate routers to make sure that the datagram fits in the underlying network packet payloads. When a datagram is fragmented into two or more datagrams, one bit of the flags field, More Fragments (MF), is used to indicate the last fragment for reassembly of the datagram at the destination. Another bit in this field, Don't Fragment (DF), is used by the source IP to indicate prohibition of fragmentation. The third bit is unused at the moment.

Fragment offset (13 bits)

This field indicates the position of the fragment in the original datagram.

Time to live (8 bits)

This field determines the maximum hops a datagram is allowed to travel. Each time a router processes a datagram, it decrements this field by one. If

the time-to-live field reaches zero, the datagram is discarded. The hop limit prevents a strayed datagram from circling the Internet forever.

Protocol (8 bits)

At the source host, many different protocols, such as TCP and UDP, use the IP protocol to send their data in IP datagrams. A mechanism is, therefore, needed at the destination to demultiplex the incoming datagrams and hand them in to the correct protocol processes. IP uses the 8-bit protocol field to uniquely identify a given protocol. TCP is identified by 6, and UDP is identified by 17.

Header checksum (16 bits)

The checksum protects the IP header against bit errors. Without the header checksum, a bit error in the header can cause incorrect routing of the IP datagram.

Source IP address (32 bits)

This field contains the source IP address. All IP network interfaces are given unique IP addresses throughout the global Internet. All IP addresses are 32 bits long.

Destination IP address (32 bits)

This field contains the IP address of the destination host of the packet.

Option (variable)

This field is not compulsory. When used, the options field contains one or more options specified by the source IP. There are four options currently defined—security, source routing, route recording, and timestamping. Because the source can specify a variable number of options, this field has a variable length.

Padding

Because the options field is variable, padding of zero bytes is used to make sure that the IP header is a multiple of 32-bit words.

2.3.4 Internet control message protocol (ICMP)

In the Internet, an IP datagram travels through many intermediate routers before being delivered to its destination. Therefore it is likely that on some

occasions, one or more routers or the destination itself may be down. If the datagram cannot be delivered to the destination because the destination is down, or if a router cannot find a route to the destination, or if the destination IP address does not exist, the routers need a mechanism to report such conditions to the hosts. Internet control message protocol (ICMP) is such a message mechanism implemented as part of the IP protocol.

Using ICMP, routers can send many different messages identifying different types of error conditions to source hosts. ICMP messages are sent by encapsulating them in IP datagrams as shown in Figure 2.5.

Figure 2.6 shows the format of a generic ICMP message. The 8-bit type field identifies the type of the message. Table 2.2 shows 15 different types of messages defined in RFC 792 [5] and RFC 1256 [6]. The 8-bit code field is used by individual message types to further specify the error condition. The 16-bit checksum field covers the entire ICMP message. The message contents and size depend on the message type.

2.4 QoS in IP

The current IP, IPv4, does not support guaranteed QoS for applications at the upper layer. The service provided by IPv4 is called "best-effort" service.

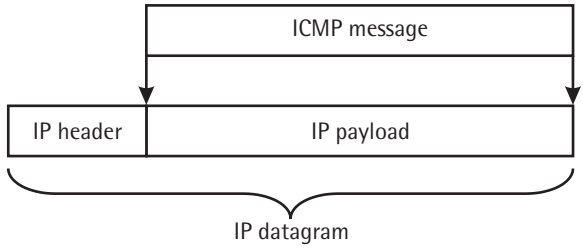


Figure 2.5 ICMP messages are encapsulated in IP datagrams.

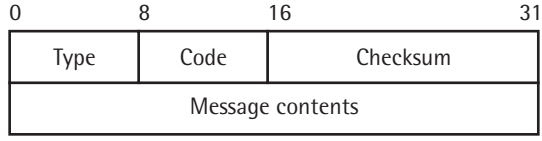


Figure 2.6 ICMP message format.

Table 2.2
ICMP Message Types

Type Number	ICMP Message Type
0	Echo reply
3	Destination unreachable
4	Source quench
5	Redirect
8	Echo request
9	Router advertisement
10	Router solicitation
11	Time (hop count) exceeded for a datagram
12	Parameter problem
13	Timestamp request
14	Timestamp reply
15	Information request
16	Information reply
17	Address mark request
18	Address mark reply

This is because the current IP routers treat all IP datagrams equally, storing them in a single first-in-first-out (FIFO) queue for sequential transmission and trying their best to deliver a datagram to the destination as soon as they can. As a result, best-effort service cannot guarantee any packet loss rate and maximum or average packet delay in the network. As a result, current IP networks are suitable primarily for data communication applications, such as file transfer and e-mail; they cannot provide adequate support for multimedia applications like voice and video, which require stringent guarantees on delay.

The IETF is now working on several new schemes to support QoS in the IP so that voice, video, and other multimedia applications can be run satisfactorily over IP networks in addition to existing data applications. One such effort has standardized a new version of the IP, IP version 6 (IPv6) [7]. The IPv6 header has a flow identifier field to uniquely identify a flow. In addition, a signaling protocol, called resource reservation protocol

(RSVP) [8], has been designed to work in conjunction with IP networks to reserve network resources in advance to guarantee QoS for a given flow.

A separate effort in the IETF has redefined the type of service field in the IPv4 header to provide priority service in IP networks. This model of supporting different services over IP networks is referred to as DiffServ [9]. It is expected that it will take a long time (5 to 10 years) to upgrade the current Internet from IPv4 to IPv6. DiffServ aims to deploy multiple service capability in the Internet for the interim period (until IPv6 and RSVP are widely deployed).

Since data communication applications that do not require QoS are expected to remain the primary sources of IP traffic in the immediate future—and probably beyond—this book does not discuss the QoS support in IP networks any further. The scope of this book is to examine and discuss the performance issues involved with running TCP-based data communication applications over ATM networks.

2.5 Summary

In TCP/IP networks, TCP is the most dominant protocol, and it is, accordingly, used by many popular applications, such as the World Wide Web, file transfer, and e-mail. TCP runs on top of IP. Typical TCP and IP packets have 20 bytes of headers and variable size payloads. TCP uses sophisticated congestion control algorithms to automatically adjust the transmission rate according to network congestion. IP provides an unreliable and connectionless service; IP datagrams may be lost or duplicated or arrive out of order at the destination. However, using the ICMP, routers can report error conditions to IP sources. All ICMP messages are sent encapsulated in standard IP datagrams.

References

[1] Comer, D. E., *Internetworking With TCP/IP, Volume I,* Third Edition, Englewood Cliffs, NJ: Prentice Hall, 1995.

[2] Stevens, W. R., *TCP/IP Illustrated, Volume 1,* Reading, MA: Addison Wesley, 1994.

[3] Postel, J., "Transmission Control Protocol," *Internet RFC 793,* IETF, September 1981.

[4] Jacobson, V., R. T. Braden, and D. A. Borman, "TCP Extensions for High Performance," *Internet RFC 1323,* IETF, May 1992.

[5] Postel, J., "Internet Control Message Protocol," *Internet RFC 792,* IETF, September 1981.

[6] Deering, S. E., ed., "ICMP Router Discovery Messages," *Internet RFC 1256,* IETF, September 1991.

[7] Deering, S., and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," *Internet RFC 2460*, IETF, December 1998.

[8] Zhang, L., S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A New Resource ReSerVation Protocol," *IEEE Network*, September 1993, vol. 7, pp. 8–18.

[9] Blake, S., D. Black, M. Carlson, E. Davis, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," *Internet RFC 2475,* IETF, December 1998.

3

ATM Networks

3.1 Introduction

The ITU has selected ATM as the network technology for realizing B-ISDN. ATM has been designed to satisfy the need to carry both real- and non-real-time data over a single network. This chapter describes the motivation behind choosing ATM to implement B-ISDN and provides an overview of ATM networks.

3.2 Why ATM networks

The following limitations of traditional telecommunications networks have driven the development of ATM and B-ISDN:

- *Service dependence:* Traditional networks were designed to support a particular service. For example, the telephone network, which was designed to carry only voice, is not suitable to carry data or video.

Similarly, cable television (CATV) networks were designed to carry TV channels and cannot be used for voice or data communication.

- *Inflexibility:* Network switches and other equipment were designed for connections of specific bandwidth (e.g., 4-kHz analog signals for analog telephony or 64-Kbps voice for narrowband ISDN). Such designs do not easily adapt to support new technologies—such as compressed voice requiring less than 64 Kbps.
- *Inefficiency:* Traditional telecommunications networks are based on circuit switching. Once a circuit is established between two end points, the bandwidth of the circuit is exclusively dedicated to these end points; other users cannot share the bandwidth even if the bandwidth is not fully utilized. This wastes network resources, especially when data services are supported, as data communication is inherently bursty and does not make continuous use of the entire circuit bandwidth.

In the late 1980s, the ITU started working on a new networking technology, known as B-ISDN, to address the above limitations of traditional networks and to meet any future service requirements without fundamentally changing the core networking infrastructure. To succeed, such a networking technology would have to incorporate the following two important features:

- *Packet switching or asynchronous transfer:* Packet switching can transfer information between two end points asynchronously without dedicated synchronous circuits. This reduces the waste of network resources, since packets from multiple sources are queued for transmission over the same link and any unused bandwidth by one source can be used by others. In addition to increasing network utilization, packet switching can support connections with a wide range of bandwidth requirements, as it is not tied to a specific data rate. This protects packet switching from becoming outmoded.
- *High speed:* Packet switches should be able to operate at a very high speed to support the high bandwidth connections of the future (e.g., HDTV).

ATM was chosen as the transfer technology for B-ISDN, because ATM was designed to be a fast packet-switching technology based on small, fixed-size packets called cells. ATM is service-independent; information from any type of service is carried in the ATM cells. Sections 3.3–3.6 present an overview of the ATM networking architecture.

3.3 ATM protocol architecture

Figure 3.1 shows the protocol architecture for ATM defined by the ITU. ATM is the common layer used by all services running over ATM networks. All information at the ATM layer is transported in 53-byte fixed-size cells. The adaptation layer is service-specific; there are different adaptation protocols for different services. The adaptation layer maps application information into ATM cells and vice versa. The physical layer supports encoding of data onto physical transmission media.

In addition to its protocol layers, the ATM protocol architecture includes three separate planes: the user plane, the control plane, and the management plane. The user plane supports the transmission of user information; the control plane provides connection controls; and the management plane coordinates activities among layers and manages network resources.

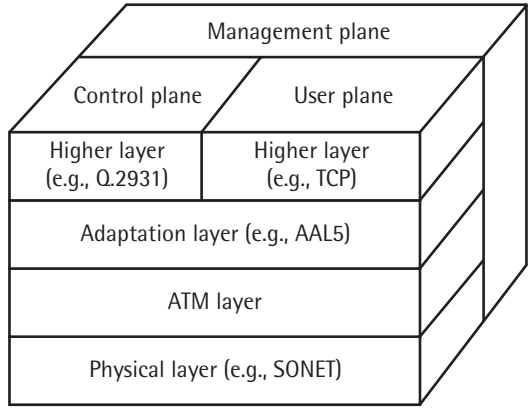


Figure 3.1 ATM protocol architecture.

3.4 How do ATM networks work?

The ATM network is based on the packet-switching technology that is used in most data communications networks. Packet switching allows multiplexing of bursty data sources to increase the bandwidth utilization of networks. Contrary to most data communications networks, which use connectionless protocols, ATM uses a connection-oriented protocol to guarantee QoS to real-time applications such as voice and video. A connection has to be set up before any data communication can take place.

ATM uses small, 53-byte–long, fixed-size cells. Since ATM is based on statistical multiplexing, cells may be lost in the network during congestion periods when the network drops cells due to insufficient buffer capacity.

3.4.1 Why are cells so short?

A network consists of a collection of network elements, usually called routers or switches, connected together by communications links. Packets arriving at the input ports of a router or switch are sent out through the output ports. In the case of congestion at an output port, packets are queued in buffers for transmission. Depending on the loss and delay priority of the packets, various queuing and scheduling disciplines are used to queue and dequeue the packets into and out of the buffers. A packet in transmission must complete transmission before another packet can be transmitted on the same port (i.e., packet transmission is nonpreemptive).

Most packet-switched networks (such as IP) are based on large variably-sized packets. When a router is transmitting a variably-sized packet, it is not possible to predict the amount of time required to finish the transmission. If a large packet is being transmitted while a packet belonging to real-time traffic such as voice is waiting for its turn, the real-time traffic will get delayed beyond the limit required to guarantee QoS to real-time applications. Figure 3.2 shows a small voice packet of 50 bytes waiting behind a large data packet of 2,000 bytes that is being transmitted. If the packets were small in size, the voice packet could have been scheduled to leave just after the completion of transmission of the current packet.

Figure 3.3 shows the data packet broken down into 40 smaller packets (data packet 1 to data packet 40) of 50 bytes each. In this case, the voice packet can be scheduled for transmission as soon as the transmission of data packet 1 is complete. Data packet 2 will be delayed, but this is not an

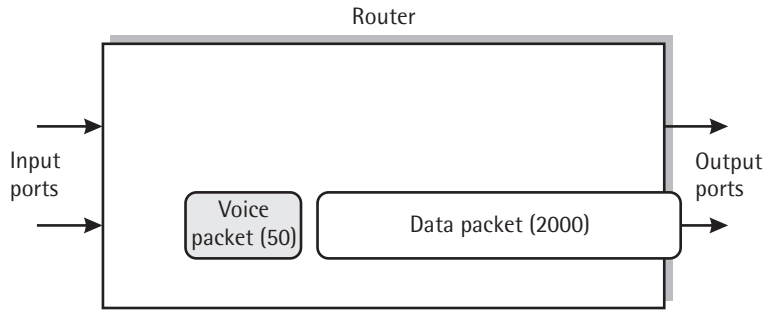


Figure 3.2 A small real-time voice packet waiting behind a large non-real-time data packet.

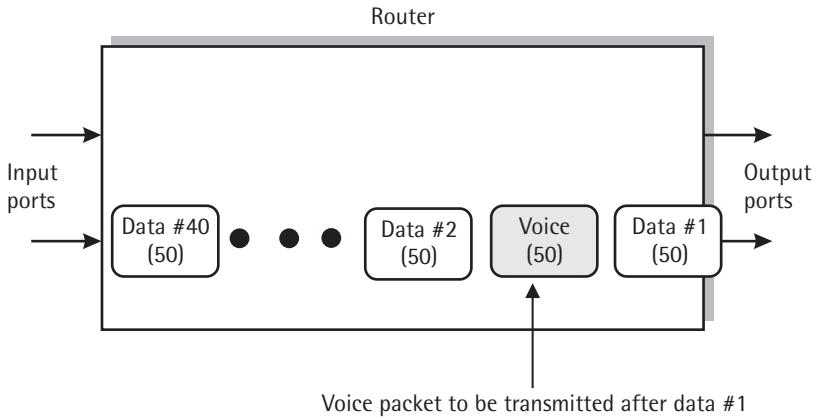


Figure 3.3 The real-time voice packet being scheduled after data packet 1.

important issue because it is a non-real-time data packet. The small size of the cells helps to give guarantees to real-time applications, and the fixed size allows the design of a simple transport network.

3.4.2 Traffic contract

When an ATM connection is established, the user and the network enter a traffic contract that has to be honored by both the user and the network for the entire duration of the connection. The traffic contract has two parts, as shown in Figure 3.4. The traffic descriptor part defines the traffic pattern the source promises not to violate, and the QoS descriptor part defines the

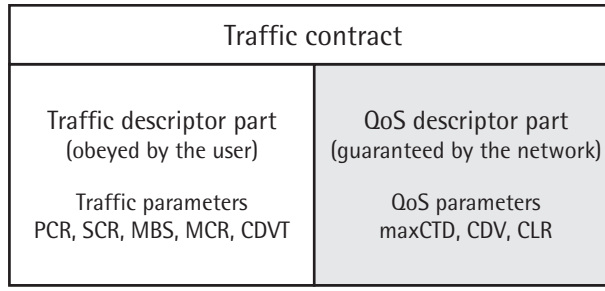


Figure 3.4 Traffic contract between the user and the ATM network.

QoS that the network promises to guarantee to the user for this connection. The traffic and QoS descriptors are defined by several parameters, described in Sections 3.4.2.1 and 3.4.2.2.

3.4.2.1 Traffic descriptors

Combinations of the following four parameters define the source traffic pattern for a given connection:

- *Peak cell rate (PCR):* The PCR defines the maximum rate at which the source can submit cells to the network over this connection.
- *Sustainable cell rate (SCR):* The SCR defines the upper bound of the average cell rate for this connection. If the average cell rate exceeds the SCR, the source would be violating the traffic contract.
- *Maximum burst size (MBS):* The MBS is the maximum number of cells that can be sent back-to-back at the peak rate.
- *Minimum cell rate (MCR):* The MCR commitment requested from the network is defined by this parameter.

In addition to the above four source traffic descriptors, another parameter that is used to define the traffic pattern of an ATM connection is the cell delay variation tolerance (CDVT). Even when a source is transmitting at the PCR, the delay between cells is not always constant at the User Network Interface (UNI). The variation in delay is caused by factors such as transmitting physical layer overhead cells and cell multiplexing. CDVT defines the maximum cell delay variance.

3.4.2.2 QoS descriptors

The following QoS descriptors were defined by the ATM Forum:

- *Maximum cell transfer delay (maxCTD):* The CTD is the time spent by a cell in the network, including any fixed delays, such as propagation and switching delays, and variable delays such as queuing delay. The maxCTD imposes an upper limit on CTD.
- *Cell delay variation (CDV):* The CDV is the difference between the best case and worst case of CTD. The best case is the fixed delay, and the worst case is the maxCTD.
- *Cell loss ratio (CLR):* The CLR is the ratio of lost cells to total cells transmitted. Although the primary cause of cell loss in ATM networks is buffer overflow at intermediate switches, this is not the exclusive cause. Cells may also be lost due to misrouting of cells.

3.4.3 ATM service classes

ATM supports five different service classes, each one providing a different level of QoS. The class of service is specified by the sources during connection setup. Depending on the required service, the network reserves appropriate resources to satisfy the QoS requirements. The following services have been defined by the ATM Forum [1]:

- The constant bit rate (CBR) service, which is similar to a leased line, is characterized by the PCR. It requires a guaranteed bandwidth and delay from the network.
- The variable bit rate (VBR) service is characterized by the PCR, the SCR, and the mean burst size. The VBR service is further divided into real-time VBR (rt-VBR) and non-real-time VBR (nrt-VBR) depending on the guarantees required from the network.
- The ABR service, which was designed for data applications, uses rate-based congestion control to manage the traffic in the network. In case of network congestion, the network uses feedback to inform the sources to reduce the data rate. An MCR is negotiated at connection setup, but the MCR may be zero. The feedback is implemented by resource management (RM) cells. ABR sources periodically send special RM cells that are marked by the switches with congestion information and the amount of bandwidth available at the switches.

The destination turns around the RM cells toward the sources. The sources, on receiving the RM cells, adjust their source rates depending on the network congestion and the amount of bandwidth available at the switches. A set of source, destination, and network rules specify the operation of ABR connections. As long as the ABR sources conform to the rate specified by the network, the loss rate of the ABR connection is very low.

- The unspecified bit rate service (UBR) does not provide the user with any bandwidth or delay guarantee. The network tries its best to carry such traffic. In case of congestion, the network drops cells and relies on the end applications to detect such losses and initiate retransmission. UBR is equivalent to the *best-effort* service provided by the IP layer in the current Internet.

Table 3.1 shows the relevance of traffic and QoS parameters to each service class.

Figure 3.5 shows the usage of a link bandwidth among various service classes during steady-state operation (i.e., a period of time without any addition or deletion of Virtual Circuits [VCs]). A constant amount of bandwidth is reserved and used by the CBR sources, followed by the variable amount of bandwidth being used by the VBR sources. There is no mechanism to slow down the data rate of the CBR or VBR connections; hence, congestion control in ATM networks is achieved by reducing the data rate of the ABR sources. It should be noted, however, that it is possible to reserve a non-zero minimum bandwidth for an ABR connection, in which case the ABR connection continues to receive at least the minimum

Table 3.1
Traffic and QoS Specifications for Different Service Classes

Service Class	Traffic Parameter	QoS Parameter
CBR	PCR	maxCTD, CDV, CLR
rt-VBR	PCR, SCR, MBS	maxCTD, CDV, CLR
nrt-VBR	PCR, SCR, MBS	CLR
ABR	PCR, MCR	CLR (network-specific; some networks may not allow the specification of a CLR)
UBR	PCR	No QoS

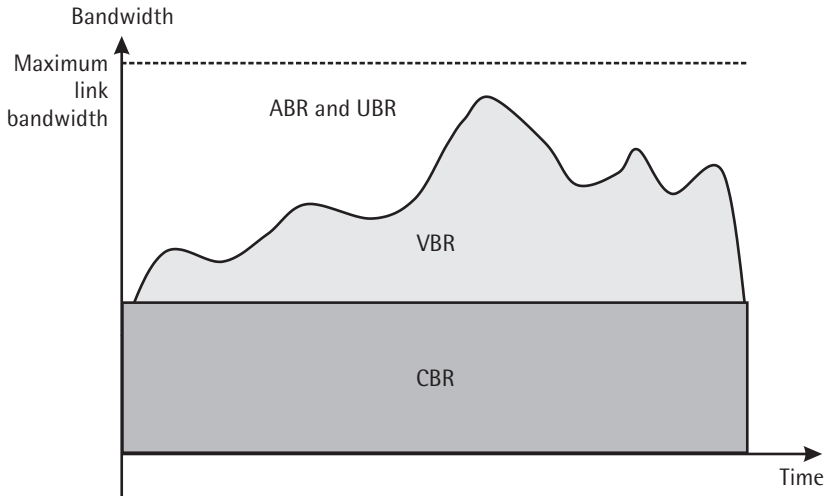


Figure 3.5 Distribution of link bandwidth among various traffic classes during steady state.

bandwidth plus any leftover bandwidth not used by the CBR and VBR sources. Any bandwidth not used by CBR, VBR, or ABR is used to transmit UBR traffic.

3.4.4 Congestion control

ATM is based on statistical multiplexing of packets. The inherent assumption is that not all users will be transmitting at their peak rate at any instant of time. However, if many users are transmitting at their peak rate during any instant of time, the network can become congested and cause cell loss within the network. Several mechanisms are used by ATM to control congestion and guarantee QoS to existing connections.

One such mechanism is connection admission control (CAC). Under this method, a source sends a connection request with the required connection parameters to the network before transmission of data. The network checks whether it has enough resources to accept the connection request without affecting the QoS of existing connections. A connection request may be denied if the network does not have enough resources to accept the connection.

A second method to control congestion is policing, whereby an ATM network monitors the traffic sent by every connection to check whether the

connection is adhering to the traffic contract that has been set up during CAC. Excess traffic can either be discarded by the network at the point of entry or can be tagged as low-priority traffic to be discarded first in the case of network congestion.

A third method of controlling congestion is to reduce the rate of traffic entering the network. In the case of network congestion, control packets are sent back to the ABR sources, ordering them to reduce their source rate. ABR service will be discussed in Chapter 5.

3.4.5 Error checking

ATM networks are designed to operate on very high-speed links that have very low rates of error. Optical fiber, which has very low bit error rates (BERs) and high bandwidth, is an example of such a link. Because they contain such high-quality links, ATM networks do not perform error correction of the payload and do not retransmit lost cells—unlike many networks that perform hop-by-hop error correction or retransmission of lost cells at the data link layer. In ATM networks, error checking of the payload or retransmission of lost cells is the responsibility of higher layers such as the end applications. Nevertheless, to prevent misrouting of cells, ATM does perform hop-by-hop error checking on the header fields.

3.5 ATM connection types

ATM is a connection-oriented network. Accordingly, it needs to set up a connection before data can be transmitted over the network. Connection setup requires a finite amount of delay before data can be transmitted. Three different types of connections can be set up depending on the user requirements.

Permanent VCs (PVCs) are set up between two points by the network operator. PVCs are more appropriate for a high volume of data transfer, and they usually stay in place for months before being torn down by the operator. A semipermanent PVC, on the other hand, can be set up and torn down more frequently as the need arises. The duration of semipermanent PVCs usually ranges from a few days to months. PVCs and semipermanent PVCs are suitable for implementing virtual leased lines over ATM networks. It is worthwhile to mention that data transfer does not encounter

any VC setup delay with PVCs, as these PVCs are set up prior to any data transfer.

For low-volume, bursty data transfers lasting only a few seconds to a few minutes, a switched VC (SVC) is set up and released by a user using signaling from the application. A finite amount of time is required to set up the circuit before data transmission can start.

PVCs do not provide a scalable solution for ATM networks. When there are a large number of sources and destinations, PVC-based solutions will require a large number of VCs to be maintained in the network. Although most early implementations of ATM used PVCs for simplicity (no signaling), future ATM networks will increasingly use SVCs to dynamically set up and release ATM connections on demand.

3.6 ATM adaptation layers

Different applications have different service requirements for factors such as timing, delay, and variability of transmission rate. The ATM adaptation layer (AAL), which resides on top of the ATM layer, allows any type of service or protocol to connect to ATM. The AAL layer accepts data from any other protocol and transmits the data in fixed size ATM cells.

Each adaptation layer is further divided into two sublayers, the segmentation and reassembly (SAR) sublayer and the convergence sublayer (CS). The CS can add more information to the data units, such as a sequence number. The higher layer protocol data units (PDU) are broken down into fixed-sized ATM cells by the SAR sublayer.

To satisfy the different service requirements of the applications, the following four AALs have been defined:

- AAL1: For CBR services, such as 64-Kbps voice;
- AAL2: For carrying VBR data, such as MPEG2 coded video;
- AAL3/4: For transporting data from connection-oriented packet switching networks.
- AAL5: For supporting connectionless data communications, such as TCP/IP traffic. Figure 3.6 shows the protocol stack for running TCP/IP applications over ATM using AAL5.

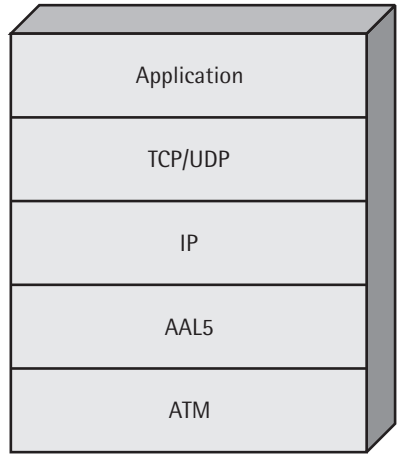


Figure 3.6 Protocol stack for TCP/IP over ATM.

AAL5 does not have any header; it only has an eight-octet trailer. Figure 3.7 shows the format and protocol fields of AAL5 PDU at the CS layer. The fields are defined as follows.

- The payload (0–65,535 octets) is a variable-length field to hold the higher layer data.
- The PAD (0–47 octets) is a variable-length field used to make the length of the AAL5 PDU a multiple of 48 octets; each of these 48 octets is then carried in one ATM cell.
- User-to-user (UU) is a single-octet field not directly used by the AAL; it is used by a higher layer for purposes like sequencing and multiplexing.

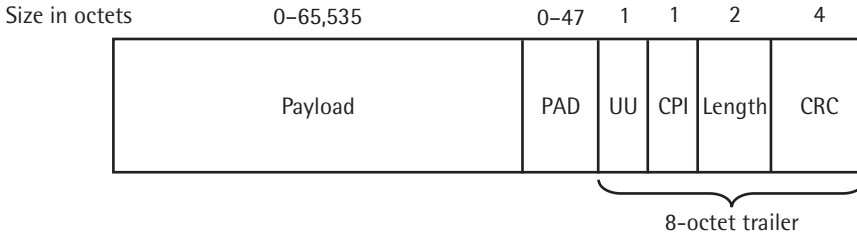


Figure 3.7 Format of AAL5 encapsulation.

- The common part indicator (CPI), another single-octet field, signals which interpretation should be used to interpret the rest of the fields in the trailer. Since only one interpretation is defined at the moment, this field is not used.
- The length (2 octets) field specifies the length of the data being carried in the payload field.
- The cyclic redundancy check (CRC) field (four octets) is used to detect bit errors in AAL5 PDU using CRC.

The AAL5 CPCS-PDU is broken down into segments of 48 octets that are carried as the payload of ATM cells. A five-octet header is added to the 48-byte payload to form an ATM cell that is then transported over the ATM network.

3.7 Physical layers

ATM is not dependent on any specific physical layer technology. The ATM Forum and the ITU, however, defined several physical layer standards for carrying ATM traffic. This section describes the framing structure of three high-speed physical layer protocols used to transmit ATM cells between two adjacent nodes [2].

3.7.1 DS3

DS3 is a digital data service provided by many carriers for data communication. The ATM Forum has specified the use of DS3 at the UNI. A physical layer convergence protocol (PLCP) is used to map the 53-byte ATM cells into the payload of a DS3 circuit as shown in Figure 3.8. A PLCP frame encapsulating ATM cells consists of 12 rows, each containing an ATM cell and four octets of overhead for such factors as alignment, synchronization, and parity. PLCP frames are carried in DS3 payloads. A PLCP frame is, therefore, transmitted every 125 microseconds (i.e., 8,000 frames per second) resulting in an effective data rate of $12 * 424 * 8{,}000$ bps = 40.704 Mbps, where the line rate of DS3 is standardized at 44.736 Mbps.

3.7.2 TAXI

TAXI is a 100-Mbps interface that was originally introduced as a physical interface for FDDI networks. TAXI is based on multimode fiber and is thus

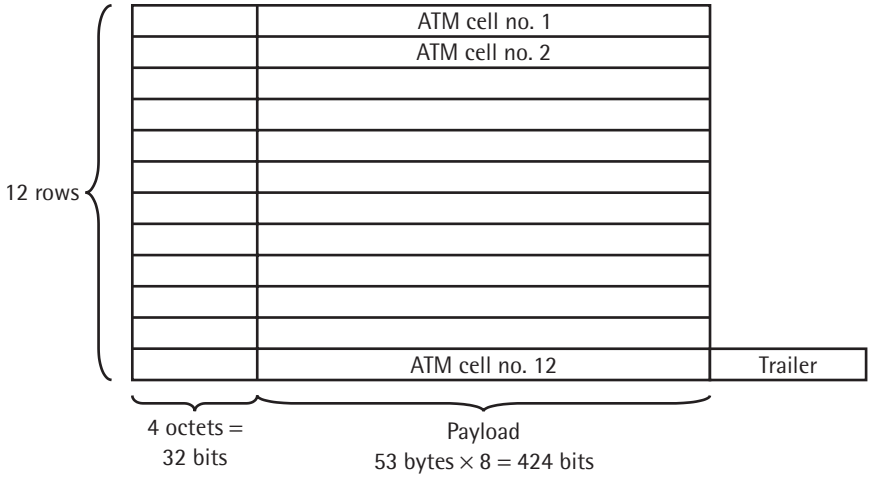


Figure 3.8 Structure of a DS3 PLCP frame.

limited to 2 km between devices. It is, therefore, unsuitable for MANs. It uses 62.5-micron optical fiber with a 125-Mbps line rate and 4B/5B encoding, which yields a bit rate of 100 Mbps.

No special framing format is defined for the TAXI interface. Instead, ATM cells are delineated by two special codes, TT, the start of cell code, and JK, the idle code. The TT and JK codes are each one byte long. One TT code is inserted at the start of each cell, and at least one JK is inserted between two adjacent cells (i.e., for back-to-back ATM cells, there will be one JK and one TT code between two ATM cells), as shown in Figure 3.9.

Because there are two extra bytes (TT and JK bytes) sent for each 53-byte ATM cell, the percentage of TAXI bandwidth available to the ATM layer is 53 * 100/55 = 96.36%. Hence for a 100-Mbps TAXI link, only 96.36 Mbps is available to ATM due to TAXI protocol overhead.

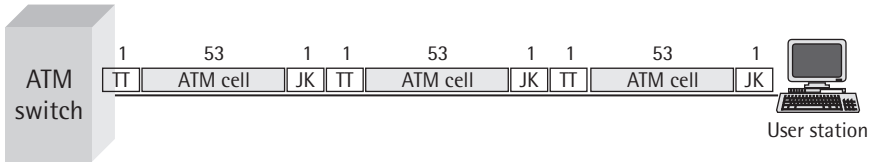


Figure 3.9 TAXI format for the transmission of back-to-back ATM cells.

3.7.3 SONET

Carriers in different countries use different, incompatible transmission layer technologies. The SONET was an effort from Bellcore (and later standardized by the ITU) to design an optical network that could be used internationally to enable communication between the networks of different countries. The ITU recommendations are referred to as the SDH.

SONET is a synchronous system that is run by a high-precision master clock. It contains only one channel and uses time division multiplexing (TDM) to carry several channels over a single channel. Several line speeds have been defined for SONET. For example, OC-1, OC-3, and OC-12 have line speeds of 51.840 Mbps, 155.520 Mbps, and 622.080 Mbps.

The size and format of a SONET frame depend on the line speed. A SONET frame running at the OC-1 speed consists of 810 bytes (nine rows of 90 bytes) as shown in Figure 3.10. The first three columns carry system management and multiplexing information such as line overhead, and the remaining 87 columns are available for user data. One of the 87 columns is used for path overhead. The path overhead is required when a number of synchronous transport signal (STS) channels from different connections are multiplexed. However, when multiple STS channels from the same connection are multiplexed, the path overhead is not required, resulting in all 87 columns being available to the user.

A SONET frame is sent every 125 microseconds, resulting in 8,000 frames per second and, at the OC-1 rate, a total data rate of 810 * 8 * 8,000 bps = 51.84 Mbps, or STS-1. Several STS-1 channels can be multiplexed on a byte basis to form higher capacity channels. For example, an

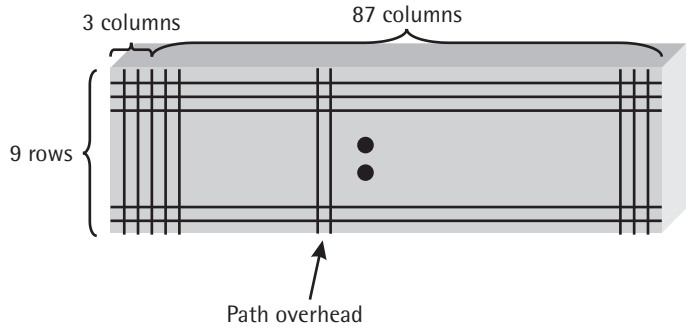


Figure 3.10 SONET frame structure for OC-1.

STS-3 or OC-3 channel with a date rate of 51.84 * 3 = 155.52 Mbps can be formed by multiplexing three STS-1 channels.

As an example, multiplexing of three STS-1 frames from different channels results in an OC-3 frame with a user data rate of 86 * 8 * 9 * 3 * 8,000 bps = 148.608 Mbps out of a raw OC-3 data rate of 90 * 8 * 9 * 3 * 8,000 bps = 155.52 Mbps. However, when three STS-1 channels from the same source are multiplexed to form an OC-3c (where the c stands for concatenated STS-1 channels from the same source) channel, the one-column path overhead is required in only the first STS-1 frame. The second and the third STS-1 frames can use all the 87 columns for user data. This gives a user data rate of (86 + 2 * 87) * 8 * 9 * 8,000 bps = 149.76 Mbps, resulting in an efficiency of 149.76/155.52 = 96.3 % for OC-3c.

For OC-12c, 12 STS-1 channels from the same connection are multiplexed, with the first STS-1 frame using the path overhead column and the remaining 11 frames using the full 87 columns for user data, resulting in an effective user data rate of (86 + 11 * 87) * 8 * 9 * 8,000 bps = 600.768 Mbps. This results in an efficiency of 600.768/12 * 51.84 = 96.57% for OC-12c.

3.8 Conclusions

This chapter presents the salient features of ATM and discusses its advantages over other networks. It describes the AAL protocols required to run various applications and protocols, such as TCP/IP, over ATM networks, concluding with detailed descriptions of three popular physical layer framing formats—DS3, TAXI, and SONET—used in ATM networks.

References

[1] ATM Forum Traffic Management Specification, version 4.1, December 1998.

[2] Prycker, M. D., *Asynchronous Transfer Mode: Solution for Broadband ISDN, Third Edition,* Englewood Cliffs, NJ: Prentice Hall, 1995.

4

TCP/IP Over ATM

4.1 Introduction

The ATM network was designed to be used as both a LAN and a WAN. Consequently, it was once envisioned that the ATM network would become a ubiquitous network that would replace most of the current networks. However, because of the large installed base of legacy LANs (e.g., Ethernet) and associated TCP/IP infrastructure and the higher prices of ATM equipment, the current trend is to continue to use legacy LANs as much as possible and to use ATM as a high-speed backbone network to interconnect legacy LANs using TCP/IP. Therefore, TCP/IP and ATM are going to coexist and internetwork in the foreseeable future. This chapter describes various configurations and architectures that have been proposed for running TCP/IP over ATM networks.

4.2 ATM deployment in TCP/IP networks

Running TCP/IP over ATM networks involves, among other things, breaking large, variably-sized TCP/IP packets into small, fixed-size ATM cells.

There are two ways to deploy TCP/IP over ATM networks—ATM to the desktop and ATM in the backbone—depending on the point in the network at which the breaking of the IP packets takes place.

The *ATM to the desktop* approach is based on connecting the hosts directly to an ATM network using ATM network interface cards, as shown in Figure 4.1. The ATM network essentially replaces the legacy LAN, and ATM is configured to operate as an emulated LAN. The segmentation of the TCP/IP packets into ATM cells takes place at the hosts. In the *ATM in the backbone* approach, the hosts in an enterprise are connected using legacy LANs, and a gateway connects the enterprise to the ATM network using a multiprotocol router, as shown in Figure 4.2. In this case, the segmentation of the TCP/IP packets into ATM cells occurs at the gateway, and the hosts are not aware of the presence of ATM.

For IP hosts or routers interfacing directly with ATM, new mechanisms are necessary to support IP traffic over ATM. Section 4.3 explains two standard mechanisms for running IP over ATM.

4.3 Running IP over ATM

There are a number of differences between legacy LANs and ATM networks. For example, legacy LANs use connectionless shared media, and hence no connection setup is required before transmitting a packet. In contrast, ATM is based on setting up point-to-point (or point-to-

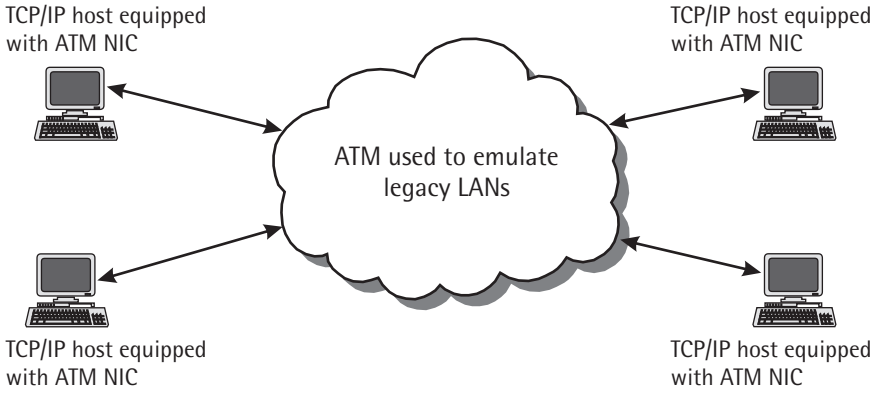


Figure 4.1 TCP/IP hosts connected to an ATM LAN using the ATM to the desktop approach.

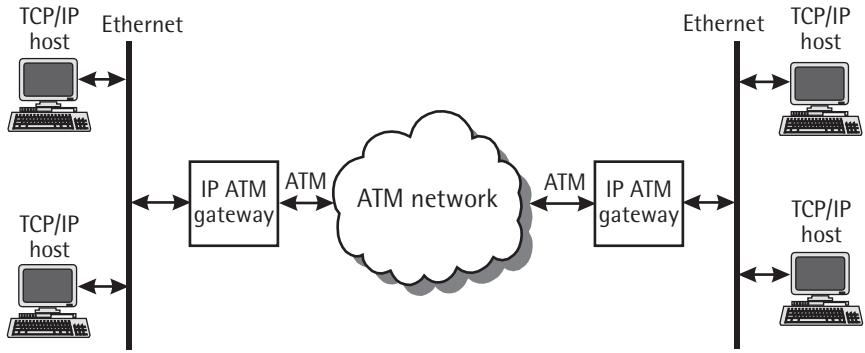


Figure 4.2 TCP/IP networking using the ATM in the backbone approach.

multipoint) connections before data can be sent over a connection. Therefore, mechanisms are necessary to run IP over ATM. The two standard approaches for running IP over ATM, LAN emulation (LANE) and classical IP over ATM, are described in Sections 4.3.1 and 4.3.2, respectively.

4.3.1 LAN emulation (LANE)

LANE was proposed by the ATM Forum [1, 2] to allow hosts on legacy LANs to *transparently interoperate* with hosts directly connected to ATM switches or ATM networks. With LANE, the ATM network effectively becomes an extension of an existing legacy network where an ATM host can be treated as a legacy host and can be accessed using a legacy MAC address. Software that runs on legacy LAN hosts also runs on ATM hosts without any changes. Figure 4.3 illustrates the concept of transparent interoperability between an existing legacy LAN (Ethernet) and a newly deployed ATM LAN using LANE.

LANE allows ATM hosts to run existing legacy LAN software without making any changes. However, since legacy LAN software uses broadcast to communicate with other hosts on the LAN, a mechanism is necessary in LANE to support broadcast over VC-based ATM networks. Sections 4.3.1.1 and 4.3.1.2 describe the two possible approaches for communication between hosts connected through an ATM LANE.

4.3.1.1 Many point-to-multipoint connections

In this approach, each host opens a separate VC connection to every other host in the emulated LAN resulting in a fully connected mesh of VCs, as

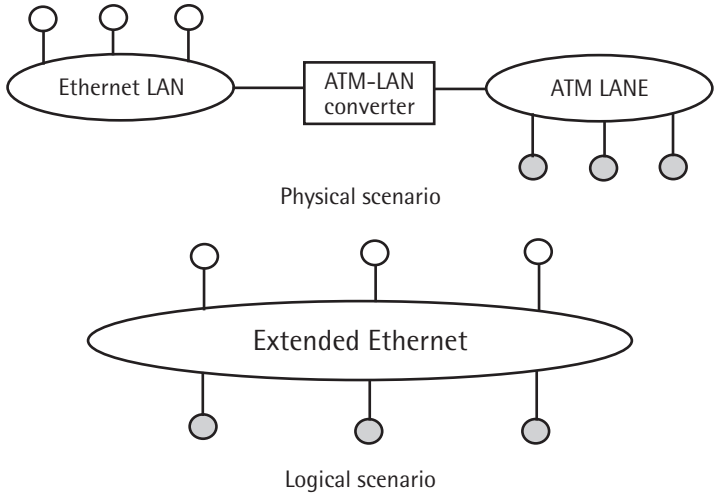


Figure 4.3 Interoperability between a legacy LAN and an ATM LAN using LANE.

shown in Figure 4.4. However, as the number of hosts in the emulated LAN increases, this method is not scalable since the number of connections that must be updated or opened every time a host joins or leaves the LAN becomes excessive.

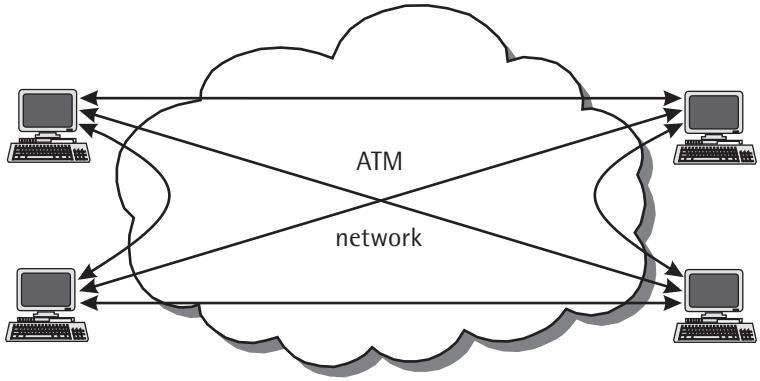


Figure 4.4 Many point-to-multipoint connections to implement an emulated LAN.

4.3.1.2 One point-to-multipoint connections

The scalability problem in the multipoint-to-multipoint approach gave rise to a second approach whereby all the hosts are connected to a multicast server as shown in Figure 4.5. A host aiming to broadcast data sends the data to the multicast server, which then sends it to all the hosts in the LAN. Here, the server has a point-to-multipoint connection, and the hosts only have point-to-point connections with the server.

The ATM Forum's LANE architecture, which consists of a number of hosts running the LANE client (LEC), uses the one point-to-multipoint approach. Each LEC in the emulated LAN is connected to three servers as shown in Figure 4.6. When a LEC joins an emulated LAN (ELAN), it sets up connections with all three servers. The functions of the three servers are described as follows.

- *Broadcast and unknown server (BUS):* A LEC that wants to broadcast a packet to all the hosts sends the packet to the BUS, which then forwards it to all the hosts. The BUS is also used when the sending LEC does not know the address of the receiving LEC.
- *LANE server (LES):* The LES is used to resolve the mapping between MAC addresses and ATM addresses. If a LEC does not have the ATM address of another LEC with which it wants to communicate, the LEC sends out a LANE address resolution protocol (LE_ARP) request to the LES, which either returns the ATM address from its

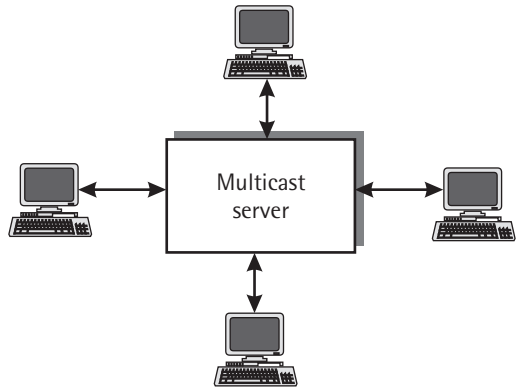


Figure 4.5 A number of hosts connected to the multicast server.

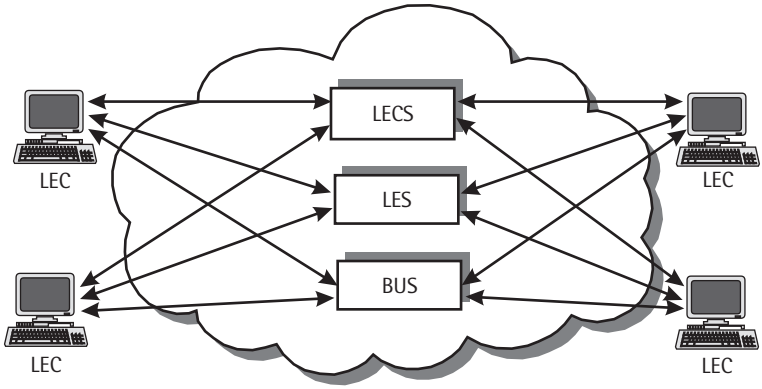


Figure 4.6 Servers required in the ATM Forum's LANE architecture.

cache or broadcasts the LE_ARP request to all LECs that have registered with the LES. There is one LES per ELAN.

- *LANE configuration server (LECS):* The LECS maintains a database of configuration information for the ELAN. For example, when a LEC joins the ELAN, the LECS informs the LEC of the ATM address of the LES with which it should register. To distribute the load, there could be a number of LECS in a distributed fashion within an ELAN. The LECS is configured by the systems administrator.

The primary advantage of LANE is that all existing protocols, such as IP, IPX, and Netbeui, can be supported readily without modification. The limitations of this approach, of course, are the lack of access to ATM signaling and the fact that there is no guarantee of QoS for the higher layers.

4.3.2 Classical IP over ATM

The classical IP over ATM method was standardized by the IETF [3] as a mechanism to run IP over ATM networks where IP treats ATM as a new technology (with no emulation of Ethernet or token ring) and has access to all ATM functions and features including the guaranteed QoS. In this architecture, the IP hosts that are connected to an ATM network form a logically independent subnet (LIS) with a single ATM address resolution protocol (ARP) server to resolve the translation between IP and ATM addresses as shown in Figure 4.7.

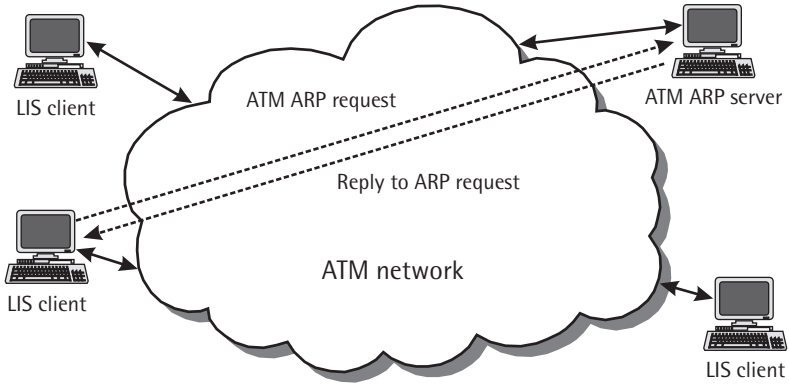


Figure 4.7 Connecting a number of hosts using classical IP over ATM.

When a new IP host (LIS client) is connected to the network, it informs the ATM ARP server of its IP and ATM addresses, which are configured by the system administrator. When a LIS client wants to determine the ATM address of a second LIS client, it sends a request to the ATM ARP server, which returns a reply to the ARP request informing the LIS client of the ATM address. The LIS client can then communicate with the second LIS client using the ATM address obtained from the ATM ARP server. If a host is not part of the LIS, the message has to be sent to a router that will have to forward the packet to the appropriate subnet. Classical IP over ATM is an IP-only solution; similar but separate solutions are needed to run IPX or Netbeui over ATM.

4.4 Encapsulating IP packets into ATM cells

Section 4.3 discusses standard architectures to connect TCP/IP hosts using an ATM network. This section shows how the IP packets are encapsulated into ATM cells so that the underlying ATM network can carry them.

Two different methods of encapsulating connectionless data over an ATM network using AAL5 were suggested in RFC 1483 [4]. The first method, called LLC/subnetwork attachment point (SNAP) encapsulation, multiplexes a number of connectionless data streams (belonging to different protocols) over a single ATM VC using a logical link control (LLC) header. The second method uses individual ATM VCs to carry packets

belonging to different protocols; hence, it is called VC-based multiplexing. Since both routers (or hosts) working at layer 3, and bridges (or LANE hosts) working at layer 2, can be connected to ATM networks, the two standard encapsulation methods use slightly different formats for carrying layer 3 (routed) and layer 2 (bridged) protocols. Sections 4.4.1 and 4.4.2 describe LLC/SNAP encapsulation and VC-based multiplexing for both routed and bridged protocols.

4.4.1 LLC/SNAP encapsulation

The LLC/SNAP multiplexing technique is used when several protocols are carried over the same ATM VC. Sections 4.4.1.1 and 4.4.1.2 discuss the AAL5 payload format for the routed and bridged protocols.

4.4.1.1 Routed protocols

To enable the destination to differentiate between the different protocols that are being carried over the same VC, the source appends a three-octet LLC header and a five-octet SNAP header; they are carried in the AAL5 payload of the transmitted data. Figure 4.8 shows the AAL5 payload for encapsulating an IP packet.

The SNAP header consists of a three-octet Organizationally Unique Identifier (OUI) and a two-octet Protocol Identifier (PID). The OUI identifies the organization that administers the meaning of the codes used to identify different protocols in the PID field.

4.4.1.2 Bridged protocols

The payload for bridged protocols differs slightly from the payload of the routed protocols described in Section 4.4.1.1. As an example, the

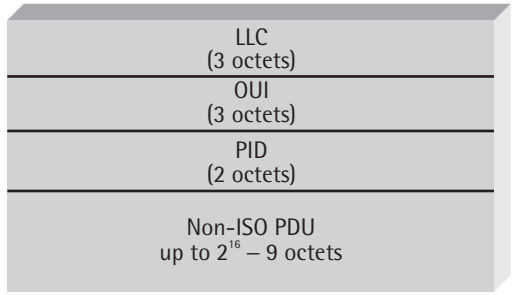


Figure 4.8 AAL5 payload for encapsulating an IP packet.

AAL5 payload for the bridged Ethernet is shown in Figure 4.9. A value of 0xAA-AA-03 in the LLC field indicates the presence of the SNAP header consisting of the OUI and PID fields. A value of 0x00-80-C2 in the OUI field represents the organizational code for the IEEE 802.1 working group. A PID value of 0x00-01 represents the presence of a LAN Frame Check Sequence (FCS) field and a bridged Ethernet protocol, whereas a PID value of 0x00-07 represents a bridged Ethernet protocol with no LAN FCS field. The LAN FCS field contains the FCS of the original PDU.

Since the PDUs of all the different protocols are carried over the same VC in the LLC/SNAP encapsulation method, the method is suitable when it is not convenient or possible to dynamically open a large number of VCs without incurring significant cost.

4.4.2 VC-based multiplexing

The VC-based multiplexing scheme is used when a large number of VCs can be opened without incurring significant cost. In the VC-based multiplexing scheme, a host opens a number of VCs to the destination, each VC being used to carry packets for a different protocol as shown in Figure 4.10. The destination host differentiates between the PDUs of the different protocols by the different VC numbers. There is no overhead (such as the LLC overhead used in the LLC multiplexing scheme) to differentiate packets from different protocols.

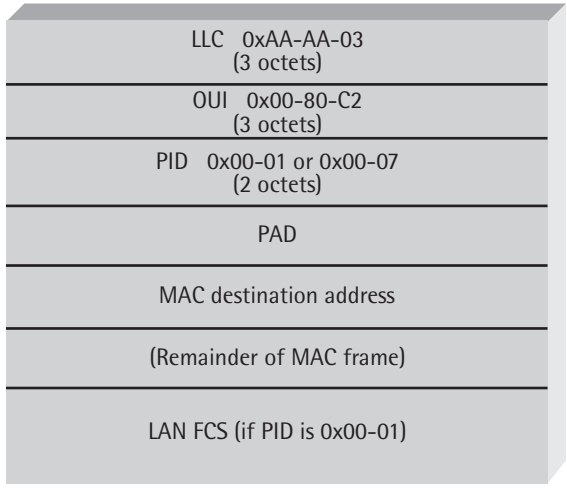


Figure 4.9 AAL5 payload for the bridged Ethernet.

The advantage of this method is that it requires minimal bandwidth to transmit data and only a small overhead to process headers. This scheme is better than the LLC multiplexing scheme when a large number of VCs can be dynamically opened very quickly without incurring much cost. It is anticipated that this scheme will prevail in the private ATM networks. Sections 4.4.2.1 and 4.4.2.2 describe techniques to implement the VC-based multiplexing scheme in the routed and bridged protocols.

4.4.2.1 Routed protocols

Since the destination does not need to differentiate between PDUs carried in a particular VC, the AAL5 payload field using VC-based multiplexing and routed PDUs will consist of only the IP PDU as shown in Figure 4.11 for IP over ATM.

4.4.2.2 Bridged protocols

When PDUs of bridged protocols are carried using the VC-based multiplexing scheme, the payload format is similar to when they are carried using the LLC encapsulation scheme except that the LLC, OUI, and PID fields are no longer required. Figure 4.12 shows the payload format for a bridged Ethernet carried using the VC-based multiplexing scheme.

4.5 Summary

This chapter discusses techniques to connect TCP/IP hosts using ATM. The first half of the chapter describes two ATM adoption approaches—ATM to

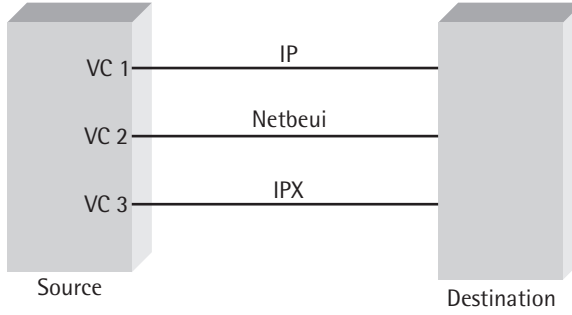


Figure 4.10 Two hosts exchanging multiprotocol data over ATM using VC-based multiplexing.

Figure 4.11 AAL5 payload format for encapsulating IP datagrams in the VC-based multiplexing scheme.

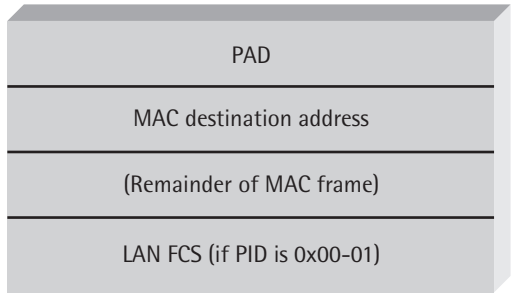


Figure 4.12 Payload for a bridged Ethernet in the VC-based multiplexing scheme.

the desktop and ATM at the backbone—and techniques to implement them. The second half of the chapter discusses techniques to break and encapsulate large variably-sized IP packets into small, fixed-size ATM cells.

References

[1] ATM Forum, "LAN Emulation Over ATM Specifications," Version 1, ATM Forum Specifications, February 1995.

[2] Sun, H., K. Huang, and L. Li, "Supporting IP on the ATM Networks: An Overview," *Computer Communications,* Vol. 21, No. 11, August 1998, pp. 1020–1029.

[3] Laubach, M., and J. Halpern, "Classical IP and ARP Over ATM," *Internet RFC 2225*, IETF, April 1998.

[4] Heinanen, J., "Multiprotocol Encapsulation Over ATM Adaptation Layer 5," *Internet RFC 1483,* IETF, July 1993.

5

Performance Issues for TCP Over ATM

5.1 Introduction

ATM networking technology promises substantial gains by allowing a mix of audio, video, and data communication over the same network. However, a number of recent studies show that certain features of ATM and the way in which TCP interacts with ATM may cause substantial degradation to TCP throughput. Unless the poor performance of TCP over ATM is addressed adequately, the gain by ATM may not be realized in future TCP/IP networks based on ATM.

Currently, the performance problem for TCP/IP over ATM networks is usually addressed using "brute-force" methods that result in over-dimensioning expensive network resources, such as link bandwidth. While over-dimensioning, the network provides an ad hoc solution to the performance problem. However, it does not provide an economic and scalable solution for the future. To solve the performance problem more efficiently, it is necessary to understand the actual sources of the performance problem, which often involves examining the complex interactions between the TCP/IP and ATM protocols.

This chapter introduces the following six performance issues for TCP/IP over ATM networks:

1. Loss of link bandwidth due to ATM protocol overhead (also known as ATM cell tax);
2. ATM cell loss;
3. ATM connection setup delay;
4. Interaction between TCP's congestion control and ATM's rate control;
5. Effect of large ATM MTU for IP over ATM;
6. Limitation of standard TCP window size for high-speed wide-area ATM connections.

Various algorithms and mechanisms addressing the first five performance issues are discussed in Chapters 6 through 10. The solution to the sixth performance problem, which is rather trivial, is discussed in this chapter.

5.2 ATM protocol overhead

A communications protocol at a given layer transmits some information of its own in addition to the data it receives from the higher layer. The protocol-related extra information is referred to as protocol overhead. Such overhead is usually appended to the head (as a header) and/or at the trail (as a trailer) of the information received from the higher layer. The headers and trailers play an integral role in providing a reliable, efficient communication service by implementing a range of functions (e.g., detecting transmission errors and routing of packets through the network). The protocol overhead is not part of the user data and is thus seen as loss of transmission bandwidth from the user's perspective.

While protocol overhead is unavoidable, an important goal of protocol design is to minimize overhead to give users better access to the available link bandwidth. Transmission efficiency, or the ratio of usable bandwidth to the total link bandwidth, measures how well protocols achieve this goal and should be considered as a performance issue for the protocol.

Transmission efficiency is particularly important for long-distance communications links, where the bandwidth is considered very expensive.

To illustrate the importance of transmission efficiency, let's consider a 100-Kbps leased line between Australia and the United States for a multinational corporation with offices in both countries. If the protocol overhead contributes 20% to the total traffic, it means that only 80 Kbps is available to the corporation for transporting useful data; the rest of the bandwidth is consumed by the communication network itself. Put another way, if the corporation needs 100 Kbps of available bandwidth, it has to over-dimension the link capacity by 25 Kbps (80% of 125 is 100). This increases the effective price of the available bandwidth, which may prove costly in circumstances such as satellite-based long-distance leased lines, which are very expensive.

To assess the transmission efficiency for TCP/IP over ATM networks, let us examine the complete ATM protocol stack, including the physical and adaptation layers, supporting the transmission of TCP/IP over ATM networks. Figure 5.1 shows the protocol stack for TCP/IP over ATM with three popular physical layer protocols: SONET (OC-3 for 155.52 Mbps

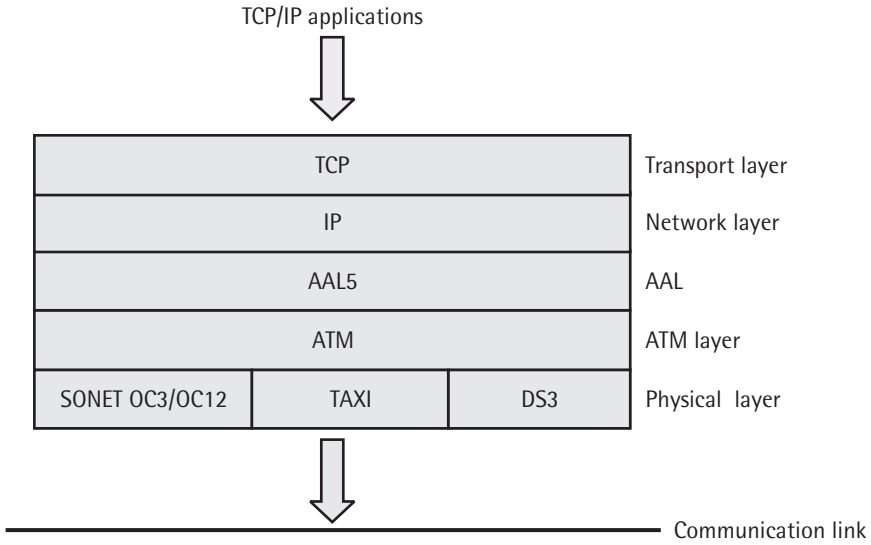


Figure 5.1 TCP/IP over ATM; protocol stack with three different selections of physical layer.

and OC-12 for 622.08 Mbps), TAXI, and DS3. TCP packets are first encapsulated into IP datagrams, which are, in turn, encapsulated into AAL5 packets and so on until the physical layer packet is finally transmitted onto the physical transmission link. Every layer in the protocol stack contributes to the overall overhead.

The way IP datagrams are encapsulated into AAL5 packets may have some effect on the efficiency calculation for IP over ATM. As mentioned in Chapter 4, RFC 1483 provides two options for encapsulating IP datagrams into AAL5 packets: IP packets can be either encapsulated directly into AAL5 packets without any LLC header (for the VC multiplexing option, which requires separate VCs for separate higher layer protocols), or an 8-byte LLC header is attached to an IP datagram before encapsulating it into an AAL5 packet (for multiplexing different higher layer protocols into a single VC).

For 40-byte IP datagrams, the smallest but most frequently found datagrams carrying TCP ACKs, encapsulation using VC multiplexing will not add any padding, as the total number of bytes in the AAL5 packet is 48 (40-byte IP datagram plus an 8-byte AAL5 trailer). However, for LLC encapsulation, the size of the AAL5 packet becomes 56 bytes without the padding and 96 after padding (96 is an integral multiple of 48). Hence, LLC encapsulation significantly reduces the transmission efficiency for IP over ATM for 40-byte IP datagrams.

Fortunately, LLC encapsulation does not have any effect on the transmission efficiency for the rest of the *commonly found datagram sizes* on the Internet. For example, the size of the AAL5 packet for carrying a 576-byte IP datagram, the default datagram size used for nonlocal IP destinations, is 624 no matter which encapsulation option is selected. This is because, with VC multiplexing, 40 bytes of padding is used, whereas with LLC encapsulation, only 32 bytes of padding is used. In both cases, the AAL5 packet becomes 624 bytes, an integral number of 48 bytes.

We now examine the transmission efficiency for TCP/IP over ATM with the three different physical layer protocols shown in Figure 5.1. Chapter 3 discusses the framing format of the three physical layer protocols, along with the format for ATM and AAL5; these formats allow us to precisely calculate the overhead and hence the transmission efficiency of each protocol layer. For example, each SONET OC-3c frame is 2,430 bytes long, including a total overhead (section, line, and path overhead) of 90 bytes. In other words, for every frame only 2,340 out of 2,430 bytes

are available to the higher layer, yielding a transmission efficiency of 2,340/2,430 or 96.3% at the SONET layer.

Because an IP datagram travels through three layers of protocols—the AAL5, the ATM, and the physical—before being transmitted onto the transmission link, we consider all overheads from physical layer to AAL5 in calculating the transmission efficiency for IP over ATM. It should be noted here that the transmission efficiencies for various applications running on top of TCP will be less than the IP efficiency presented here, as TCP and IP have their own overheads.

Table 5.1 shows the transmission efficiencies of each layer in carrying data from the immediate upper layer. The first four rows show the efficiencies of four different physical layers for ATM networks. The efficiency of the ATM layer is shown in the fifth row. The sixth row shows the efficiency of AAL5 for four frequently found sizes of IP datagrams: Forty-four–byte datagrams, which carry TCP FIN and SYN packets, appear on the Internet very frequently [1]; 576 bytes is the default datagram size for nonlocal IP destinations [2]; 1,500-byte datagrams are typically generated by Ethernet-attached hosts (Ethernet frames have a maximum payload of 1,500 bytes); and 9,180 is the default datagram size [3] for IP sources directly connected to an ATM network. As can be seen, except for AAL5, all protocols have a fixed efficiency; AAL5 has a variable efficiency that depends on the size of IP datagrams. This is because of the variable length padding (between 0 to 47 bytes) added at AAL5 to produce an integral number of fixed-size ATM payloads (see Section 3.6). It is worth mentioning here that both VC multiplexing and LLC encapsulation have the same efficiency for all four sizes of IP datagrams considered in Table 5.1.

The overheads of individual layers in a layered architecture with one protocol stacked on top of another (Figure 5.1) have a compound effect on the total transmission efficiency achieved by IP over ATM. For example, 44-byte datagrams with a choice of SONET OC-3c link will have 96.3% of the link bandwidth available at the ATM layer, 87.22% at the AAL5 layer (90.57% of 96.3 is 87.22), and finally 39.97% at the IP layer (45.83% of 87.22 is 39.97). The transmission efficiencies achieved by different IP datagram sizes for a range of physical layer selections are calculated similarly and shown in Table 5.2. The first column shows the previously discussed four frequently found IP datagram sizes. For each datagram size, the third column shows the transmission efficiency (percentage of link bandwidth available to IP) for four different physical layers (column 2). The raw link

Table 5.1
Transmission Efficiency of Various Protocols

Protocol	Transmission Efficiency
SONET OC-3c	96.3%
SONET OC-12c	96.57%
TAXI	96.36%
DS-3	90.99%
ATM	90.57%
AAL5	45.83% (for 44-byte IP datagrams) 92.31% (for 576-byte IP datagrams) 97.66% (for 1,500-byte IP datagrams) 99.61% (for 9,180-byte IP datagrams)

bandwidth and the effective throughput at the IP layer—after taking the overheads at all layers—are shown in fourth and fifth columns, respectively. The values in the fifth column are simply obtained by multiplying column 4 by column 3.

A glance through Table 5.2 reveals that the choice of physical layer has little bearing on the efficiency. However, efficiency depends significantly on the size of the IP datagram. For small IP datagrams, the efficiency is significantly low; it is only around 40% for 44-byte IP datagrams irrespective of the choice of physical layer. Efficiency improves dramatically for larger datagrams; 80% efficiency is achieved with 576-byte datagrams, and an even greater level is achieved for 9,180-byte datagrams.

Because the efficiency of ATM depends primarily on the size of IP datagrams, it is important to look at the distribution of IP datagram sizes on the Internet to assess the overall efficiency of IP over ATM. Recent studies on Internet traffic patterns showed peaks at packet sizes of 44, 576, and 1,500 bytes [1]. An analysis [1]—taking into account the distribution of datagram size—of over 16 billion observed datagrams (over 5 trillion bytes of IP data) on a high-speed commercial Internet backbone link revealed that IP over ATM has an overall efficiency of 80% at the ATM layer if LLC encapsulation is used, and this is without considering the overhead at the physical layer. The efficiency would improve to 84.47% if VC multiplexing were used for IP encapsulation over AAL5.

Table 5.2
Transmission Efficiency for IP Over ATM Networks

IP Datagram	Physical Layer	Efficiency	Raw Link Bandwidth	Max. Effective bandwidth
44-byte	SONET OC-3c	39.97%	155.52 Mbps	62.16 Mbps
	SONET OC-12c	40.08%	622.08 Mbps	249.33 Mbps
	TAXI	40.00%	100 Mbps	40 Mbps
	DS-3	37.77%	44.736 Mbps	16.90 Mbps
576-byte	SONET OC-3c	80.51%	155.52 Mbps	125.21 Mbps
	SONET OC-12c	80.74%	622.08 Mbps	502.27 Mbps
	TAXI	80.56%	100 Mbps	80.56 Mbps
	DS-3	76.07%	44.736 Mbps	34.03 Mbps
1,500-byte	SONET OC-3c	85.18%	155.52 Mbps	132.47 Mbps
	SONET OC-12c	85.42%	622.08 Mbps	531.38 Mbps
	TAXI	85.23%	100 Mbps	85.23 Mbps
	DS-3	80.48%	44.736 Mbps	36.00 Mbps
9,180-byte	SONET OC-3c	86.88%	155.52 Mbps	135.12 Mbps
	SONET OC-12c	87.12%	622.08 Mbps	541.96 Mbps
	TAXI	86.93%	100 Mbps	86.93 Mbps
	DS-3	82.09%	44.736 Mbps	36.72 Mbps

This high overhead of ATM is often referred to as the *cell tax*. Because of this high cell tax, some ISPs are already considering alternatives to ATM (e.g., IP directly over SONET) [4]. However, there is a strong group that supports IP over ATM even if it means paying a high cell tax. The argument used by the supporters of ATM is that the cell tax is a small price to pay for the bandwidth savings that ATM can achieve by statistically multiplexing a large number of bursty Internet traffic sources onto the same channel. In contrast, SONET is a synchronous transmission mechanism; once a circuit is established, no bandwidth from this circuit can be shared by other circuits even if the circuit remains under-utilized. Therefore, it is likely that ATM will continue to serve as a backbone carrier for Internet traffic for years to come.

Since protocol overhead is an issue that is considered at the design phase of a protocol, it may seem that there is little that can be done now to reduce the ATM cell tax. In most cases, in fact, the cell tax is merely accepted as part of the ATM network, and nothing is done to reduce it. Fortunately, however—for the ISPs deploying ATM in their Internet backbone—research shows that the cell tax can be greatly reduced by using some standard compression techniques. Chapter 6 details these compression techniques and their effect on the ATM cell tax.

5.3 Effect of ATM cell loss on TCP/IP

One of the main advantages of ATM is the efficient utilization of link bandwidth through the *statistical multiplexing* of a large number of variable-rate (bursty) input traffic sources onto a single output link. With statistical multiplexing, the capacity of the output link is less than the sum of the peak rates of all ATM connections but higher than the total average rate of all connections. The statistical gain is a factor by which the sum of peak rates exceeds the output link capacity. ATM allows statistical multiplexing at the cell level inside the ATM switches.

A direct consequence of statistical multiplexing is the potential for congestion at the output link when the total input rate exceeds the capacity of the output link. The technique used to address the momentary congestion in a statistical multiplexer is to buffer the excess cells until the total input rate drops below the link capacity. However, because of the finite capacity, buffers may overflow if the congestion persists for a long time. Incoming cells are lost (discarded by the ATM switch) when buffers overflow.

In 1995, a simulation study by Romanow [5] revealed that ATM cell loss has a detrimental effect on the throughput of TCP connections running over ATM networks; the throughput can be as low as 34% of maximum achievable throughput. To appreciate the magnitude of the effect of cell loss on TCP performance, one must understand the entire process of TCP/IP communication over ATM networks. Accordingly, the remainder of this section illustrates the dynamics of TCP traffic over ATM networks and the effect of cell loss on TCP throughput.

A TCP segment is encapsulated in an IP datagram, which, in turn, is encapsulated in an AAL5 packet, which is finally segmented and transmitted in multiple ATM cells through the ATM network. Therefore, one

TCP segment travels in a set of cells; the longer the TCP segment, the larger the set is. However, since the switching units within the ATM network deal with individual cells, the network is not aware that a group of cells carry the same TCP segment. Consequently, any cell from the set of cells carrying the TCP segment may be dropped independently in the ATM network if congestion occurs.

It is easy for the ATM network to drop a cell when buffer overflows, but the recovery process from a cell loss is a painful one: Because of the potentially high overheads at high speed, the ATM layer does not implement retransmission of lost cells; the job of cell loss recovery is left to the upper layers. Cell loss is eventually detected by the CRC error-detection process at the receiving AAL5 layer when an attempt is made to reassemble the original AAL5 packet from a partial set of cells. Although AAL5 implements an error-detection mechanism, it does not support any *error correction* (neither forward error correction to reconstruct the missing cells nor any backward error correction to retransmit the lost cells). Instead, upon detection of a CRC error, AAL5 simply discards the *entire* AAL5 packet. Consequently, the receiving TCP does not receive the corresponding segment (the segment that was traveling in the discarded AAL5 packet) and hence does not generate an acknowledgment for that segment.

In the absence of any acknowledgment, the sending TCP eventually times out and retransmits the entire segment, which results in retransmission of many cells that already made it to the destination in the previous attempt. Such retransmissions of correctly received cells only add insult to injury for the congested ATM switches, as a significant portion of ATM switch bandwidth is used by these redundant cell transmissions at a time when bandwidth is scarce. As a result, it takes longer than normal to recover from such congestion in the ATM network.

Another interesting phenomenon that intensifies the effect of cell loss is the way AAL5 reassembles the cells at the receiver. The last cell generated by an AAL5 packet is marked by the sender by setting a bit in the cell header. At the reassembly process in the receiver, this last cell is used to delimit one AAL5 packet from the next. If the last cell is lost, the reassembly process continues to assemble cells into an AAL5 packet until the last cell of the next AAL5 packet is received or a maximum threshold is reached. This corrupts not only the AAL5 packet whose last cell was lost but also the next AAL5 packet. Therefore, a single cell loss can potentially cause the loss of more than one AAL5 packet and may simultaneously affect the throughput

of multiple TCP connections. (Each AAL5 packet may carry TCP segments from different TCP connections.)

When cell loss occurs in an ATM network during congestion, TCP throughput is affected in several ways. First, because some ATM bandwidth is used by redundant cell transmissions, the effective link bandwidth available for TCP is reduced. The amount of bandwidth lost to transmission of redundant cells depends on a number of factors, including TCP segment size and the number of TCP connections multiplexed on the link. For a large number of multiplexed TCP connections with large segments, the loss of link bandwidth becomes quite high.

Another reason for low TCP throughput in the presence of cell loss is the way TCP reacts to congestion. When the retransmit timer goes off, TCP takes it as a signal for network congestion and starts a *slow-start* process; in other words, it retransmits the segment for which the timer expired and stops any further transmission (even if there are application data ready for transmission) until it receives acknowledgment for the retransmitted segment. For long-distance connections, the delay in receiving an acknowledgment is usually quite high (the delay in the congested switch also contributes to the total delay for receiving the acknowledgment) resulting in reduced TCP throughput. If cells are dropped frequently, TCP enters the slow-start process frequently causing significant loss of throughput. When congestion builds up, acknowledgment packets also get lost, making the situation even worse. It should be noted, however, that TCP's retransmit timeout and slow-start process are not specific to ATM networks—the same effects will occur whenever congestion and packet loss are experienced no matter what protocols are used at lower layers.

The size of buffers at the ATM switch has a critical impact on the cell loss rate for TCP over ATM. For small buffers, cells are dropped at a high rate causing significant loss of throughput for TCP. Chapter 7, which discusses measures to address the cell loss problem, revisits the subject of buffer dimensioning as an approach to improve TCP performance.

Romanow [5] evaluated the effect of cell loss on the performance of TCP by simulating a number of realistic configurations (e.g., different combinations of TCP segment size and switch buffer size) for 10 TCP connections multiplexed onto a single ATM switch link. The study reported that for certain configurations, the TCP throughput achieved was as low as 34% of the maximum possible throughput.

After Romanow's classic report on the phenomenal effect of cell loss on TCP performance, a number of measures have been suggested to address the issue of cell loss and its effect on TCP performance. These measures include provisioning large buffers at the ATM switches (for UBR service), providing congestion feedback to traffic sources (for ABR service), using forward error correction at the AAL layer to recover from the lost cells, and implementing packet-based discarding in the ATM switches. Chapter 7 discusses these measures.

5.4 ATM connection setup delay

ATM is a *connection-oriented* networking technology; an ATM-level connection between source and destination must be set up first *before* any data can be transferred. The connection is released at the end of the data transfer phase. This is similar to the voice telephone network, where we first dial the telephone number of the person with whom we wish to speak, speak only after we get a connection, and then hang up when we finish our conversation. The difference between telephone and ATM networks is that the connections in ATM networks are not dedicated physical circuits; they are only logical circuits, or VCs. Many VCs are multiplexed onto the same channel in the network for efficient use of bandwidth.

Figure 5.2 illustrates the data transfer procedure for ATM networks and the timing of various events. The source that wants to set up a connection sends a setup request to the network. The request travels from switch to switch until it reaches the destination. If the network has enough capacity, an accept signal is sent back to the source. The source then transmits data, and at the end of data transmission sends a signal to release (tear down) the VC. The processing delays in the switches and the propagation delays are also shown in Figure 5.2.

Setting up an end-to-end VC is a complex task that may involve more than one switch in the ATM network as well as the source and the destination terminals. The CAC functions in the switches need to examine the current load in the network and determine whether the available network resources (e.g., the buffer space and link bandwidth) are sufficient to guarantee the service required by the connection. Therefore, the processing delay of the VC setup request and the propagation delay from the source

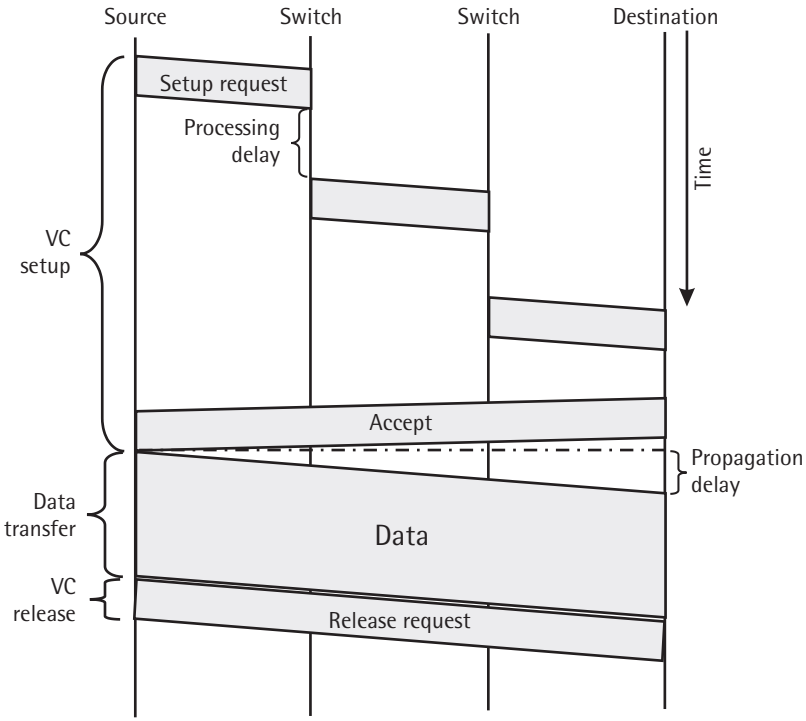


Figure 5.2 VC setup, data transfer, and VC release in ATM networks.

to destination may lead to significant connection setup delay in ATM networks.

The connection setup delay in ATM networks may affect the performance of TCP over ATM. TCP is a transport layer protocol, while ATM serves as a layer 2 protocol in the TCP/IP over ATM model (see Figure 5.1). Because of this layering distance, it is not easy to have a one-to-one mapping between TCP-level connections and ATM-level connections. It may also not be desirable to align the ATM-level connection setup and release with the ones at the TCP level, as for many interactive applications (e.g., TELNET), a TCP connection transmits short messages with relatively long silence periods between them. Maintaining an ATM-level connection for the lifetime of such a TCP connection would result in waste of network resources. To minimize the waste of network resources occupied by idle ATM connections, it is desirable to release an ATM connection after

transmitting a message and to set up another connection when the next message arrives. Therefore, during the lifetime of a single TCP connection, it is possible to have many ATM connection setups and releases. Since ATM connection setup delay could be significant, it could increase the network delay experienced by short, interactive data transfers.

Figure 5.3 illustrates the dynamics of ATM connection setup and release for transporting short, interactive TELNET messages, where ATM connections are set up each time a TELNET message is transmitted by the TCP and released after the message is transmitted. In this case, each TELNET message experiences an additional delay, equivalent to ATM connection setup delay, at the IP-ATM interface. Since performance of interactive applications like TELNET is measured by the *average message delay,* the impact of ATM connection setup is directly proportional to the connection setup delay.

The impact of ATM connection setup on the performance of TCP over ATM can be reduced by carefully managing the connection setup and release at the IP-ATM network interface. For example, instead of releasing the ATM connection after transmitting every message, the connection could be maintained for a while in anticipation of another message arrival in the near future. However, since the ATM layer cannot predict the arrival time of the next message at the TCP layer, the task of determining the holding time of an ATM connection is nontrivial. Fortunately, some newly proposed queuing models may help to determine an appropriate holding time. Chapter 8 details the management of ATM connection setup and release.

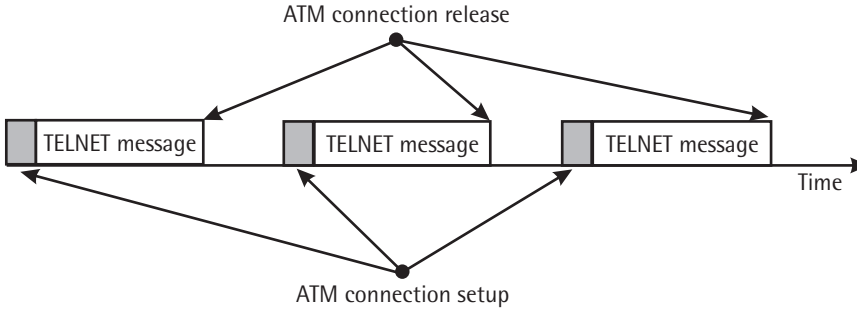


Figure 5.3 The dynamics of ATM connection setup and release in transporting short, interactive TELNET messages. Each TELNET message experiences an additional delay equivalent to the ATM connection setup delay.

The effect of VC setup delay is only relevant in the SVC environment where VCs are set up and released dynamically using signaling. However, many ATM network service providers use PVCs to support connectivity between source and destination(s). With PVCs, the VC is always there, and hence IP packets do not experience any VC setup delay in ATM networks.

PVC-based solutions, however, are not scalable for large networks with many sources and destinations and are not efficient in terms of resource utilization, since some resources are tied up even if the VC is not being used. Therefore, SVC is expected to be a more common solution for IP over ATM in the future, especially for large networks.

5.5 Effect of ABR rate control on TCP performance

As mentioned in Chapter 3, UBR and ABR are the two ATM services designed to support traditional TCP/IP-based data communications. The primary difference between UBR and ABR is that ABR provides congestion feedback to users and dynamically controls users' transmission rates to keep the ATM network congestion-free, while UBR does not provide any congestion feedback to users. UBR relies on end-to-end congestion-control techniques typically implemented within transport layer protocols (e.g., TCP). Since ABR provides good congestion control, cell loss in the ATM network can be kept to a minimum.

Since the ATM Forum released the complete specification for ABR in 1996 [6], ATM switch and network interface card (NIC) vendors have been increasingly supporting ABR in their products. It is expected that in the near future many TCP-over-ATM implementations will be based on the ABR service. It is, therefore, very important to study the performance of TCP over ABR.

In the internetworking environment, legacy networks (e.g., Ethernet) are traditionally interconnected via constant bandwidth leased lines. With the emergence of ATM, these leased lines are being replaced by the ATM ABR connections, as shown in Figure 5.4. This section discusses the end-to-end TCP performance issues in such an internetworking environment.

In the network configuration of Figure 5.4, there are two independent control loops working together; one is the ABR rate control loop bounded by the two edge routers at both ends of the ATM network, and the other is the end-to-end TCP congestion control loop working between the

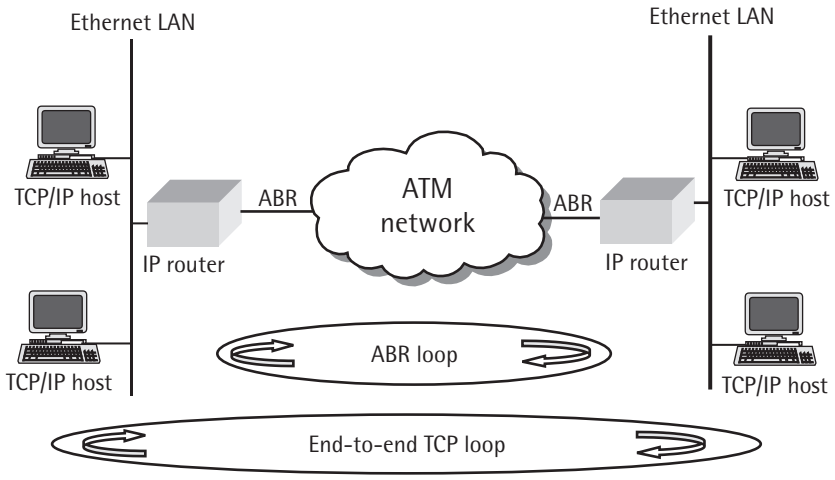


Figure 5.4 Two remote Ethernet LANs interconnected via ATM ABR service.

source and destination TCP hosts. The sole purpose of the ABR loop is to eliminate congestion within the ATM network by dynamically controlling the transmission rates of the routers. Note that the ABR rate-control feedback does not reach the TCP hosts. The objective of the TCP congestion control loop is to control the transmission of the TCP hosts according to the congestion experienced anywhere in the end-to-end path, including any congestion in the routers or in the ATM network.

Researchers argue [7–9] that the interference between these two independent control loops may actually degrade the end-to-end performance of TCP. The net effect of ABR control is to simply shift any congestion from within the ATM network to the edge routers, as the ABR control simply throttles the transmission rate of the router if congestion builds up in the ATM network. However, due to the lack of any backward (from the router to the TCP source host) congestion feedback, the TCP source host cannot learn *immediately* the rate of throttling at the router. TCP segments, which are now transmitted by the router at the new lower rate, will cause the destination TCP host to generate acknowledgments at the new rate, which, in turn, will eventually control the rate of source TCP host. Therefore, the source TCP now has to wait almost a round-trip time before learning about the congestion in the ATM network. A direct consequence of this feedback delay is that a TCP source host can momentarily

overwhelm a router (while waiting to learn about the congestion in the router), causing packet loss and TCP performance degradation, if the router bandwidth is suddenly throttled by the ABR control.

Therefore, although ABR control has proved to be very effective in controlling congestion in the ATM network, end-to-end performance of TCP over ABR in the internetworking environment (IP subnets interconnected via the ATM backbone) may be worse than leased lines or UBR connections. With leased lines and UBR connections, the transmission rates of the routers are not explicitly controlled by any external control mechanism. It was shown in [8] that in certain cases, UBR outperforms ABR in carrying TCP data.

Because the main performance problem for TCP over ABR arises from packet loss at the edge routers, we can address the problem by either implementing very large buffers at the routers or implementing intelligent a algorithms at the routers to act as a bridge between the two control loops and thereby achieve a more effective end-to-end control loop. The former approach is not scalable and will increase the delay and delay variance in the routers, which may affect other delay-sensitive traffic sharing the ABR link. The latter approach has been coined end-to-end traffic management by the ATM Forum [8]. Chapter 9 discusses a couple of end-to-end traffic management mechanisms that may significantly improve the performance of TCP over ATM ABR service in the IP-ATM internetworking environment.

Another major hurdle for achieving good TCP performance over ABR is the correct tuning of a large number of ABR parameters for a given networking environment. A comprehensive simulation study by Fang and Lin [10] revealed that the performance of TCP over ABR is very sensitive to many ABR parameter settings; TCP performs poorly for the "wrong" settings of these parameters.

Finally, one needs to consider TCP performance over *binary-mode* and *explicit-rate* ABR services. Binary-mode ABR provides only a *bang-bang* control; the sources are asked to either increase or decrease their rates. Explicit-rate ABR provides a much finer control by providing the exact allowed rate to the sources. The simulation study by Saito et al. [11] showed that better TCP performance is achieved over explicit-rate ABR than over binary-mode ABR. Since the vendors of ATM switches are increasingly supporting the explicit-rate option for ABR control, it is expected that future ABR connections will be based on explicit-rate

control and provide better TCP performance than today's binary-mode ABR connections.

5.6 Large ATM MTU and TCP deadlock

In packet-switching networks, data is carried in finite-size packets. When IP traffic is transported over a certain network (e.g., Ethernet), an IP datagram is encapsulated in the payload of the underlying network packet. The MTU of a given network is the maximum payload size of the network packet. Therefore, for TCP/IP communication over a given network, MTU refers to the maximum IP datagram size that can be allowed over that network.

Since ATM cells have a very small payload (only 48 bytes) for TCP/IP over ATM, AAL5 is used in between IP and ATM (see Section 5.1); IP datagrams are first encapsulated in the payload of AAL5 packets, which are later segmented into 48-byte chunks to fit in ATM cells. AAL5 has a payload large enough to hold a 65,535-byte IP datagram. Therefore, it is possible to send very large IP datagrams over ATM networks. However, to be in line with SMDS networks, a predecessor of ATM networks, the IETF has selected the default MTU for IP over ATM as 9,180 bytes [3]. Table 5.3 compares the ATM MTU with other networks and reveals that ATM has a much higher MTU than the traditional networks.

Since TCP segments are encapsulated into IP payloads, the MTU imposes a limit on the MSS for the TCP. TCP and IP together typically have 40 bytes of header. (IP has 20 bytes, and TCP has 20 bytes of header.) Therefore, the TCP MSS is simply 40 bytes less than the MTU. Hence,

Table 5.3
Default MTU for Various Networks

Network	Default MTU (in bytes)
X.25	576
Ethernet	1,500
FDDI	4,352
Token ring (IEEE 802.5)	4,464
ATM	9,180

for TCP/IP over ATM, the TCP MSS is 9,140 bytes. In other words, for TCP/IP communications over ATM networks, TCP is allowed to transmit 9,140-byte segments. Since the effect of the ATM cell tax decreases as the TCP segment size increases—see Table 5.2 for the effect of datagram size on bandwidth efficiency—many data communications applications (e.g., file transfer) try to avoid using segment sizes smaller than the MSS.

The remainder of this section describes *TCP deadlock*, an unfortunate phenomenon that occurs during a TCP connection when the sender waits for an acknowledgment from the receiver before it can send further data, and the receiver waits for more data from the sender before it can generate an acknowledgment. The probability of the occurrence of TCP deadlock increases with a larger MSS. Since TCP over ATM has a large MSS, it has a very high chance of causing TCP deadlock and, hence, TCP performance degradation.

The TCP deadlock problem for ATM networks was discovered and reported independently by Comer and Lin [12] and Moldeklev and Gunningberg [13] in 1995. Carefully conducted experiments of TCP/IP over an ATM LAN (two SPARC stations running SunOS connected through a Fore System ATM switch) revealed that the TCP deadlock phenomenon decreased TCP performance by 99% from expected throughput for bulk data transfers and yielded a response time one order of magnitude longer for short communications. This phenomenon had not been experienced previously with other LAN technologies, such as Ethernet.

The events that lead to TCP deadlock depend upon a number of complex operating system and TCP protocol implementation peculiarities. Therefore, this section attempts to provide only an overview of the TCP deadlock problem and its impact on TCP performance; Chapter 10 explains in detail the cause of TCP deadlock and the role of MSS in causing the deadlock and presents several alternative solutions to address the deadlock problem and improve TCP throughput.

To provide an overview of the deadlock problem, we first need to explain the concepts of send-and-receive buffers at the sender and receiver and the acknowledgment procedure followed by the receiver. The sending TCP sends data from the send buffer and leaves any unacknowledged data in the send buffer. This reduces available space in the send buffer until acknowledgments are received, and the previously sent data is cleared from the buffer. Similarly, the receive buffer at the receiving TCP stores

any received data until acknowledgments are generated, and the data are passed to the application.

There would not have been any problem if the receiving TCP generated an explicit acknowledgment for every segment it received. However, to reduce the processing load both at the receiver and at the sender, the receiving TCP generates an acknowledgment only if it receives data that is at least 35% of the receive buffer or two MSSs in size. To prevent long delays in generating acknowledgments, the receiver also sets an *acknowledgment timer* when it receives new data. If enough data are not received during the timeout period, the timer expires, triggering an acknowledgment for the data received so far.

Because of the delayed acknowledgment policy at the receiver, the TCP connection may enter a deadlock when the available space in the send buffer is smaller than the MSS, preventing the sender from transmitting another segment, and when the data received and stored in the receive buffer is smaller than both 35% of the total receive buffer size and two MSSs size. The deadlock is eventually broken by the expiry of the acknowledgment timer at the receiver. Upon expiry of this timer, the receiver generates an acknowledgment for the sender, and the sender clears previously sent data from the send buffer (upon receiving the acknowledgment), increasing the available space in the send buffer—the send buffer now has space for at least one MSS—and resumes transmission once again. However, the events that caused the first deadlock simply repeat, causing another deadlock shortly after the transmission resumes. Therefore, the TCP connection goes through cycles of two states:

- *Deadlock state:* Where the sender cannot send data, although there is application data ready and waiting to be transmitted. The sender is waiting for acknowledgment from the receiver to create more space in the send buffer to be able to transmit another MSS, and the receiver is waiting for more data from the sender to generate an acknowledgment.
- *Transmission state:* Where the available space in the send buffer is at least one MSS in size, and the sender transmits one or more segments.

Figure 5.5 shows the state transition diagram for a TCP connection experiencing cyclic deadlocks. For a TCP connection with d ms deadlocks

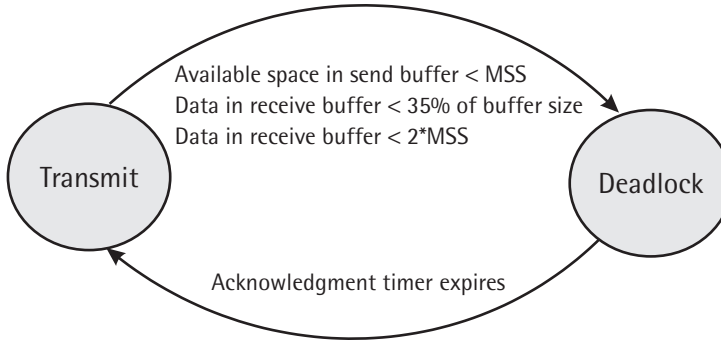


Figure 5.5 State transition diagram for cyclic TCP deadlock.

and t ms of transmission times between deadlocks, the throughput in a fraction of link bandwidth is calculated as $t/(t + d)$. Since the transmission periods between deadlocks are quite short—usually only a few segments can be transmitted before a deadlock—compared to the deadlock periods, such cyclic deadlock dramatically reduces TCP throughput.

5.7 TCP over high-delay–bandwidth ATM links

The window size field in the TCP header (see Chapter 2) has 16 bits, allowing a maximum window size of 2^{16} or 64 KB. Although a 64-KB window size seems large enough, it may become a TCP performance bottleneck for high-speed wide area ATM links with long propagation delays. For example, consider a 155-Mbps transcontinental ATM link with a 100-ms round-trip delay. For this high delay-bandwidth link, it takes 100 ms to receive an ACK from the receiver. Within this time, 1.94 MB can be sent. Therefore, in this case, a 64-KB window size will cause TCP to wait for acknowledgment and effectively prevent it from achieving 100% link utilization. The situation gets worse when the link speed and/or the distance is increased.

To increase link utilization over such high-speed ATM links, one could use TCP's window scale option (see Chapter 2), which allows a maximum window size of one GB.

It should be noted here that TCP performance limitation due to the 64-KB window size restriction is not unique to ATM links. It occurs in any

high-speed long-distance communications links, such as a satellite link with DS3 data rate.

5.8 Summary

Increasing use of ATM networks in recent years to support high-speed Internet and intranet connections has made the performance of TCP/IP over ATM an important area of research. Several performance issues need to be addressed to maximize the benefits of increasing investment in ATM networks to carry Internet traffic.

ATM carries all traffic in 48-byte payloads of 53-byte cells, introducing a hefty 10% protocol overhead or *cell tax* at the ATM layer alone. Another source of protocol overhead is due to padding in the AAL. Such protocol overheads reduce the usable bandwidth or transmission efficiency of the physical link. Network designers must take the transmission efficiency into account when dimensioning the bandwidth of ATM-based Internet connections. Compressing TCP/IP and/or ATM headers can significantly improve transmission efficiency for TCP/IP over ATM.

Cells may be lost in the ATM networks due to buffer overflow during congestion. Such cell loss has a detrimental effect on the performance of TCP. There are several ways to address the cell loss problem, including implementing large buffers in the ATM switches, setting up intelligent cell-discarding mechanisms in the ATM switches, and instituting a rate-control mechanism based on congestion feedback to adjust the source rates according to the network status.

Since ATM is a connection-oriented network, data transfer can only take place after a connection is set up. Due to the complexity involved in processing connection setup requests, it is expected that ATM connection setup delay will be nonnegligible. The performance of interactive communication may be significantly degraded if an ATM-level connection needs to be set up for every short message. ATM connections need to be managed wisely at the IP-ATM interface to minimize the effect of ATM connection setup delay on the performance of Internet applications.

Although ABR guarantees very low cell loss inside an ATM network, it does so at the price of increased congestion at the IP-ATM routers. This is because the ABR rate control mechanism simply shifts the congestion from within the ATM network to the edge of the network. The interactions

between ABR's control and TCP's end-to-end congestion control has a negative impact on the performance of TCP. End-to-end traffic management should be considered for IP-ATM internetworks to support better interaction between ABR and TCP.

An interesting performance issue, known as TCP deadlock, arises from the use of a large MTU for IP over ATM; the sender waits for an acknowledgment from the receiver before it can send further data and the receiver waits for more data from the sender before it can generate an acknowledgment. The probability of TCP deadlock occurring during a TCP connection increases with a larger MTU and hence a larger TCP MSS. TCP deadlocks dramatically reduce TCP throughput. Accordingly, one must appropriately configure various TCP implementation options to avoid TCP deadlock.

References

[1] Thompson, K., G. J. Miller, and R. Wilder, "Wide Area Traffic Patterns and Characteristics," *IEEE Network*, Vol. 11, No. 6, November/December 1997, pp. 10–23.

[2] Comer, D. E., and D. E. Stevens, *Internetworking with TCP/IP, Vol II*, Englewood Cliffs, NJ: Prentice Hall, 1994.

[3] Atkinson, R., "Default IP MTU for Use Over ATM AAL5," *Internet RFC 1626*, IETF, May 1994.

[4] Manchester, J., et al, "IP Over SONET," *IEEE Communications Magazine*, Vol. 36, No. 5, 1998, pp. 136–142.

[5] Romanow, A., and S. Floyd, "Dynamics of TCP Traffic Over ATM Networks," *IEEE Journal on Selected Areas of Communications*, Vol. 13, No. 4, 1995, pp. 633–641.

[6] "Traffic Management Specification Version 4.0," *ATM Forum*, April 1996.

[7] Newman, P., "Data Over ATM: Standardization of Flow Control," *Proc. SuperCon*, Santa Clara, CA, Jan. 1996.

[8] Jagannath, S., and N. Yin, "End-to-End TCP Performance in IP/ATM Internetworks," *ATM Forum Contribution 96-1711*, December 1996.

[9] Hassan, M., "Impact of Variable Bandwidth on the Performance of Data Communications Over ATM-ABR–Based LAN Interconnection," *Proc. 20th Australasian Computer Science Conference*, Sydney, February 5–7, 1997, pp. 422–429.

[10] Fang, C., and A. Lin, "TCP Performance in ATM Networks: ABR Parameter Tuning and ABR/UBR Comparisons," *Proc. IEEE Singapore International Conference on Networks*, April 1997.

[11] Saito, H., et al, "Performance Issues in Public ABR Service," *IEEE Communications Magazine*, Vol. 34, No. 11, 1996, pp. 40–48.

[12] Comer, D. E., and J. C. Lin, "TCP Buffering and Performance Over an ATM Network," *Internetworking: Research and Experience*, Vol. 6, 1995, pp. 1–13.

[13] Moldeklev, K., and P. Gunningberg, "How a Large ATM MTU Causes Deadlocks in TCP Data Transfers," *IEEE/ACM Transactions on Networking*, Vol. 3, No. 4, 1995, pp. 409–422.

6

Reducing ATM Cell Tax

6.1 Introduction

ATM cell tax refers to the protocol overhead in carrying IP traffic over ATM. As mentioned in Section 5.2, an ISP would incur an approximately 20% cell tax in carrying the Internet traffic typically seen in today's ISP backbones. This would reduce bandwidth efficiency (use of link bandwidth in carrying IP data) by 20%.

An abundance of small IP datagrams (between 40 and 44 bytes) that carry TCP acknowledgments and other control information are the primary contributors to such a large ATM cell tax. After being encapsulated in LLC and AAL5 packets, such small datagrams become slightly larger than the payload (48 bytes) of one ATM cell, and are hence carried over two ATM cells, the second one filled primarily with padding.

One way to reduce the cell tax is to bypass LLC encapsulation (in a method known as VC-based multiplexing), which helps to fit small IP datagrams in single ATM cells. Another way of reducing the cell tax is to actually reduce the IP datagram size by compressing the TCP/IP header. A small IP datagram with a compressed TCP/IP header will fit in one ATM cell, thereby eliminating many bytes of padding.

This chapter discusses ways in which the ATM cell tax can be reduced using VC-based multiplexing and TCP/IP header compression, both of which are standard methods for carrying TCP/IP over ATM. In addition, we describe a nonstandard method—the cells-in-frames method, which aims to reduce the ATM cell tax by sending only one ATM cell header in a frame of many ATM payloads.

6.2 VC-based multiplexing

As explained in Chapter 4, RFC 1483 [1] proposes two encapsulation methods for carrying IP datagrams over AAL5. LLC encapsulation, the typical method used by most ISPs, adds eight octets of LLC/SNAP headers in front of an IP datagram before encapsulating the IP datagram into an AAL5 packet. While the eight-octet header in LLC encapsulation allows multiple protocols (such as IP and IPX) to be carried over the same ATM VC, it imposes a hefty protocol overhead tax for small IP datagrams. Figure 6.1 shows that with LLC encapsulation, a 40-byte IP datagram does not fit in the payload of one ATM cell; a second ATM cell is needed, although it is mostly filled with padding. In this case, out of 106 bytes (two

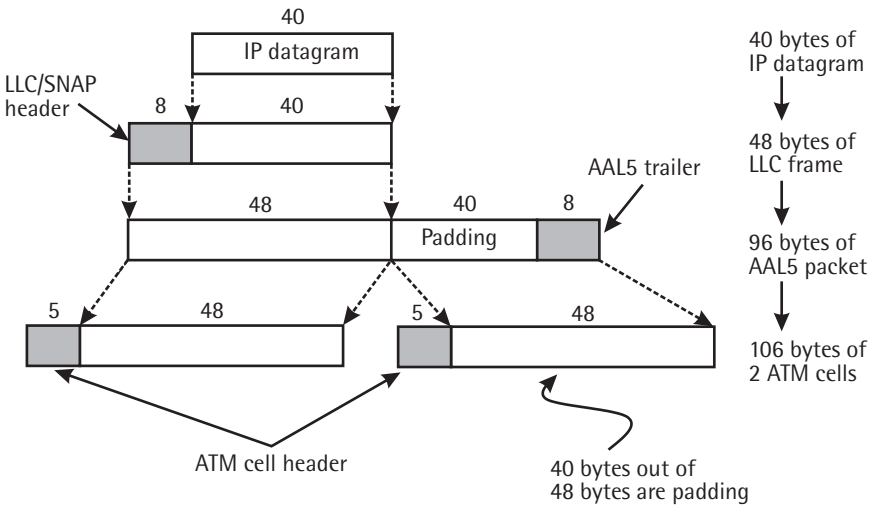


Figure 6.1 LLC encapsulation needs two ATM cells to carry a 40-byte IP datagram.

ATM cells = 2 × 53 bytes) sent, 66 bytes are ATM-related overhead. For 40-byte IP datagrams, which are found frequently in Internet traffic, LLC encapsulation, therefore, imposes a cell tax of 62% (66 * 100/106)!

With VC-based multiplexing, each protocol is carried over a separate ATM VC; the ATM VC implicitly carries the protocol information, eliminating the need for eight bytes of LLC/SNAP header. Although VC multiplexing only saves eight bytes of header for IP over AAL5 encapsulation, it has a dramatic impact on the overall savings of ATM cell tax for small IP datagrams. Now a 40-byte IP datagram can be carried in a single ATM cell (without any padding) as shown in Figure 6.2. For 40-byte IP datagrams, VC-based multiplexing imposes a cell tax of only 24% (eight bytes AAL5 and five bytes ATM overhead out of 53 bytes), 38% less tax than LLC encapsulation.

However, VC-based multiplexing has certain disadvantages that might limit its use in some environments. First, in many early deployments of ATM networks, the end-to-end connections between two routers are configured as a single ATM PVC. VC-based multiplexing is difficult, if not implausible, with such PVC configurations, as the PVC needs to be (manually) reconfigured each time a different protocol is used. Second, it may be too costly to open too many VCs—one for each protocol—if there is a charge component relating to the simultaneously active VCs. However, where dynamic creation (SVC) of a large number of VCs is fast and economical, the ATM cell tax can be reduced significantly by opting for VC-based multiplexing.

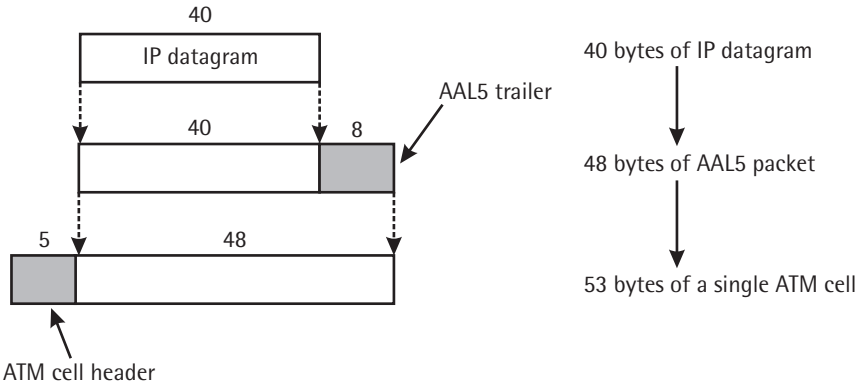


Figure 6.2 VC-based multiplexing needs only one ATM cell to carry a 40-byte IP datagram.

6.3 TCP/IP header compression

It should be clear by now that small IP datagrams (near 40 bytes) contribute to ATM cell tax much more significantly than the large datagrams. The main reason why small datagrams cause a high percentage of overhead is because they are neither small enough (after LLC/SNAP and AAL5 capsulation) to fit inside one ATM cell, nor large enough to fill up two ATM cells. The second cell carries mostly padding data (see Figure 6.1).

TCP/IP datagrams almost always (unless some optional header fields are carried) carry 20 bytes of TCP header and 20 bytes of IP header. The combined TCP/IP header is thus 40 bytes in most IP datagrams. A 40-byte header proves too big (after LLC and AAL encapsulation) for small IP datagrams to fit into a single ATM cell. One way to reduce the ATM cell tax for small IP datagrams is to compress the 40-byte TCP/IP header to fit inside single ATM cells.

In 1990, Jacobson proposed a TCP/IP header compression technique in RFC 1144 [2] that claims to be capable of reducing 40-byte TCP/IP headers to a minimum of five bytes (on average)! With such a compression ratio, small IP datagrams will fit in a single ATM cell, significantly reducing the ATM cell tax for small datagrams. Sections 6.3.1 and 6.3.2 describe the TCP/IP header compression technique presented in RFC 1144 and the impact of RFC 1144 on the ATM cell tax.

6.3.1 RFC 1144

Interestingly, RFC 1144 was not written for high-speed links like ATM; it was written almost a decade ago with the goal of improving TCP/IP performance over low-speed serial links, referring to dial-up modem connections of speeds between 300 and 19,200 bps. The main motivation behind TCP/IP header compression was to improve the response time for several popular interactive applications, such as TELNET, rlogin, and xterm.

For interactive remote terminal applications, every single key stroke (one byte) is usually sent in one 41-byte IP datagram that contains 40 bytes of TCP/IP header information. A keystroke also needs to be echoed by the remote machine. Therefore, for every character typed on the keyboard, 82 (41 + 41) bytes need to be transmitted over the slow link. A link speed of 300 bps (not an uncommon modem speed in 1990) will take over two seconds ($82 \times 8/300 = 2.19$) to display an echoed character (where

human studies show that for interactive applications, a response time above 200 ms is considered "bad")!

RFC 1144 presents a TCP/IP header compression technique to reduce the 40-byte TCP/IP header to a minimum of five bytes. With RFC 1144, a 41-byte IP datagram is reduced to a six-byte datagram. Now the serial link needs to transmit only 12 bytes for every character typed for interactive remote terminal applications. The response time for a 300-bps link reduces from more than two seconds to merely 40 ms (12/300 = 0.04 s)! Next, we describe the TCP/IP compression algorithm.

In a given TCP connection, the TCP/IP headers of consecutive datagrams carry redundant information since half of the header bytes usually do not change. Figure 6.3 shows the redundant header bytes (those that do not change) in shade. While it is easy to see why fields like protocol version and source/destination address do not change in consecutive datagrams, an explanation is in order for some specific fields. DF, MF, and fragment offset in the IP header are not relevant for small datagrams that fit in the underlying network frame, but they may be used for large datagrams that are fragmented by intermediate routers. Usually, the TCP/IP source goes through a minimum MTU discovery process and always send datagrams no larger than the minimum MTU in the end-to-end path to prevent

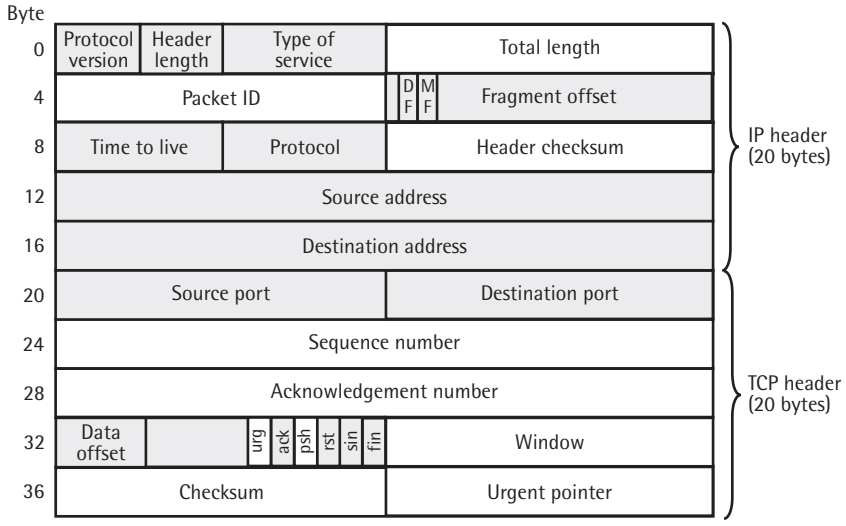


Figure 6.3 Redundant bytes/fields (in shade) in TCP/IP header.

datagram fragmentation. The SYN and FYN bits in the TCP header are only valid for the first and the last packet of a TCP connection and hence are not relevant for consecutive datagrams in the middle of a connection.

It can be seen (in Figure 6.3) that 20 bytes out of 40 bytes of TCP/IP header are redundant. Therefore, if the sender and receiver keep track of active TCP connections, and the receiver keeps a copy of the header of the first packet from this connection, the sender can easily achieve compression of almost a factor of two by sending a one-byte *connection identifier* along with the 20 bytes that change and letting the receiver fill in the 20 fixed bytes from the saved header. The TCP connection identifier in this case resembles the VC identifier (VCI) in ATM networks.

Out of the remaining 20 bytes, another four bytes can be saved, two bytes for the total length field (bytes 2 and 3) and two bytes for the IP header checksum field (bytes 10 and 11). The total length field becomes redundant when the link level protocol has a length field to tell the receiver the length of the message. Since AAL5 has a length field, the total length field is really redundant for IP over ATM. The header checksum field can be omitted too, as it is not practically checking anything useful after the omission of most of the fields in the IP header.

Omission of the total length and header checksum fields leaves 16 bytes of header to send; this includes six fields: packet ID, sequence number, acknowledgment number, window, checksum, and urgent pointer. Fortunately, not all of these six fields change at the same time; some may change only in the forward direction (sender to receiver), while others change in the backward direction (receiver to sender). For example, during an FTP session, only the packet ID, sequence number, and checksum change in the forward direction, and only the packet ID, acknowledgment number, checksum, and, possibly (if there is a window update), window change in the backward direction. With a copy of the header of the last packet sent, the sender can figure out which fields have changed and send only a bit-mask indicating which fields changed followed by the changing fields.

If the sender sends only the fields that change, the above scheme reduces the average header size to about 10 bytes. Reduction beyond 10 bytes is possible by sending only the *changed bytes* (fractions of the whole fields) in the changing fields rather than sending the changing fields in their entirety. For example, the sequence number field is incremented by the number of bytes in the next IP datagram; the increment will be less than 2^{16} (the maximum IP datagram size is 64 KB), which will change only

two bytes in the four-byte sequence number field. By sending only the *changing bytes*, it is possible to save another two bytes for the sequence number field.

Similarly one can save another few bytes from other fields where all bytes in the field do not change. It was claimed that with the above scheme, the average header size reduces to only five bytes [2]!

Figure 6.4 shows the format of a compressed TCP/IP header. There is a bitmask in the beginning of the header identifying the fields that actually changed. A bit is set in the bitmask if the associated field has changed. A connection number, which uniquely identifies a TCP connection, follows the bitmask. A 16-bit original TCP checksum is carried in the third field; then for each bit set in the bitmask, the amount of the associated field is changed. Optional fields controlled by the mask, are enclosed in dashed lines in Figure 6.4. The bit "P" in the bitmask is different than the others. It is a copy of the PUSH bit from the TCP header.

6.3.2 How RFC 1144 reduces the ATM cell tax

With the above TCP/IP header compressed from 40 bytes to five bytes, IP datagrams up to 67 bytes in length fit into a single ATM cell, as illustrated in Figure 6.5.

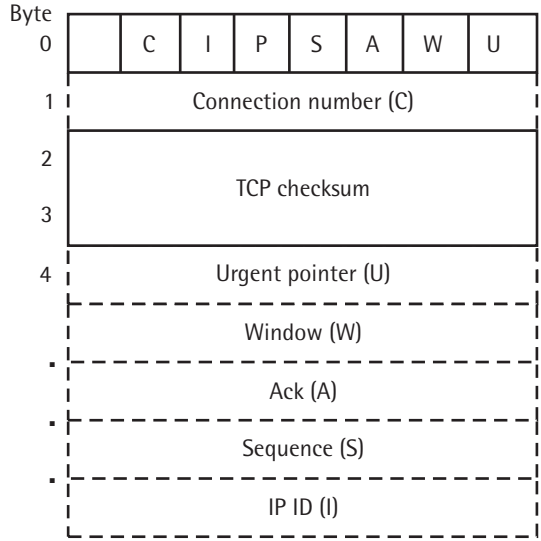


Figure 6.4 The header of a compressed TCP/IP datagram.

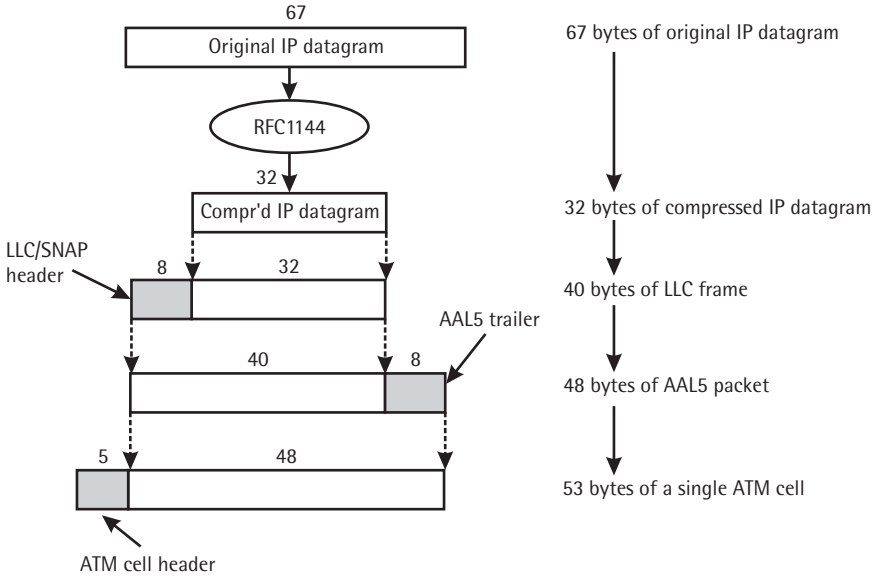


Figure 6.5 With RFC1144 compression, even a 67-byte IP datagram fits in a single ATM cell.

Since 40-byte IP datagrams constitute a large percentage of all IP datagrams in typical Internet traffic, let's examine how RFC 1144 reduces the cell tax for 40-byte IP datagrams. As shown in Figure 6.5, up to 67-byte IP datagrams can be carried in a single ATM cell using RFC 1144. Therefore, with RFC 1144 compression, a 40-byte IP datagram is transmitted in a single ATM cell of 53 bytes, yielding a cell tax of only 24%. In contrast, without RFC 1144 compression, a 40-byte IP datagram is carried in two ATM cells (see Figure 6.1.); 106 bytes are transmitted at the ATM layer for a 40-byte IP datagram, incurring a cell tax of 62%. Thus, for 40-byte IP datagrams, RFC 1144 has the same effect on cell tax as VC multiplexing.

Tax savings will be proportionately lower for larger IP datagrams. To compute the overall tax savings for typical Internet traffic containing different sizes of IP datagrams, one would need to consider the distribution of all IP datagram sizes in the traffic. Such overall tax savings are computed below in Section 6.4.

It needs to be emphasized here that the dramatic compression from 40 bytes to five bytes comes at a cost: The compression up to 21 bytes is quite easy as the sender only needs to send a one-byte connection identifier and the eight predefined fields (20 bytes) that are expected to change (see Figure 6.3). (Whether the fields actually change or not is not a concern.) However, to achieve a compression down to five bytes (on average) as shown in Figure 6.5, the compression complexity increases significantly for the sender; the sender now has to compare each of the eight fields that are expected to change with the ones in the previous packet to determine whether a field has actually changed or not. Therefore, it was noted in [2] that some hosts (e.g., low-end PCs) may not have enough processor or memory resources to implement RFC 1144 compression.

In light of the complexity involved in achieving a 40:5 compression for TCP/IP headers, it is necessary to reexamine whether a five-byte compression is desirable. While five-byte compression does have significance for TCP/IP over slow serial links using point-to-point protocol (PPP), (PPP has a variable payload that varies according to the IP datagram size it encapsulates), compression beyond 21 bytes may not be significant for TCP/IP over ATM networks. This is because ATM has a fixed payload (48 bytes); as long as an IP datagram (together with 16 bytes of AAL5 trailer and LLC/SNAP header) fits inside a single ATM cell, the actual size of the compressed datagram is not relevant, since the remaining space in the ATM cell will be filled with padding anyway.

A 21-byte compression will allow up to 52-byte IP datagrams (compared to 67-byte datagrams for five-byte compression) to fit inside a single ATM cell. Since the percentage of IP datagrams (in typical Internet traffic) between 52 bytes and 67 bytes is negligible, five-byte compression may not provide any significant benefit over 21-byte compression. From now on, we will use RFC 1144(5B) and RFC 1144(21B) to refer to five-byte compression and 21-byte compression, respectively.

Since 40-byte IP datagrams are the most frequent ones, let's examine the effects of RFC 1144(21B) and RFC 1144(5B) on 40-byte IP datagrams. Figure 6.6 illustrates the encapsulation of a 40-byte IP datagram using RFC 1144(21B) and RFC 1144(5B) compression. As Figure 6.6 shows, there is no difference in ATM cell tax (both incur 24% tax); the difference is only in the amount of padding carried in the ATM cell.

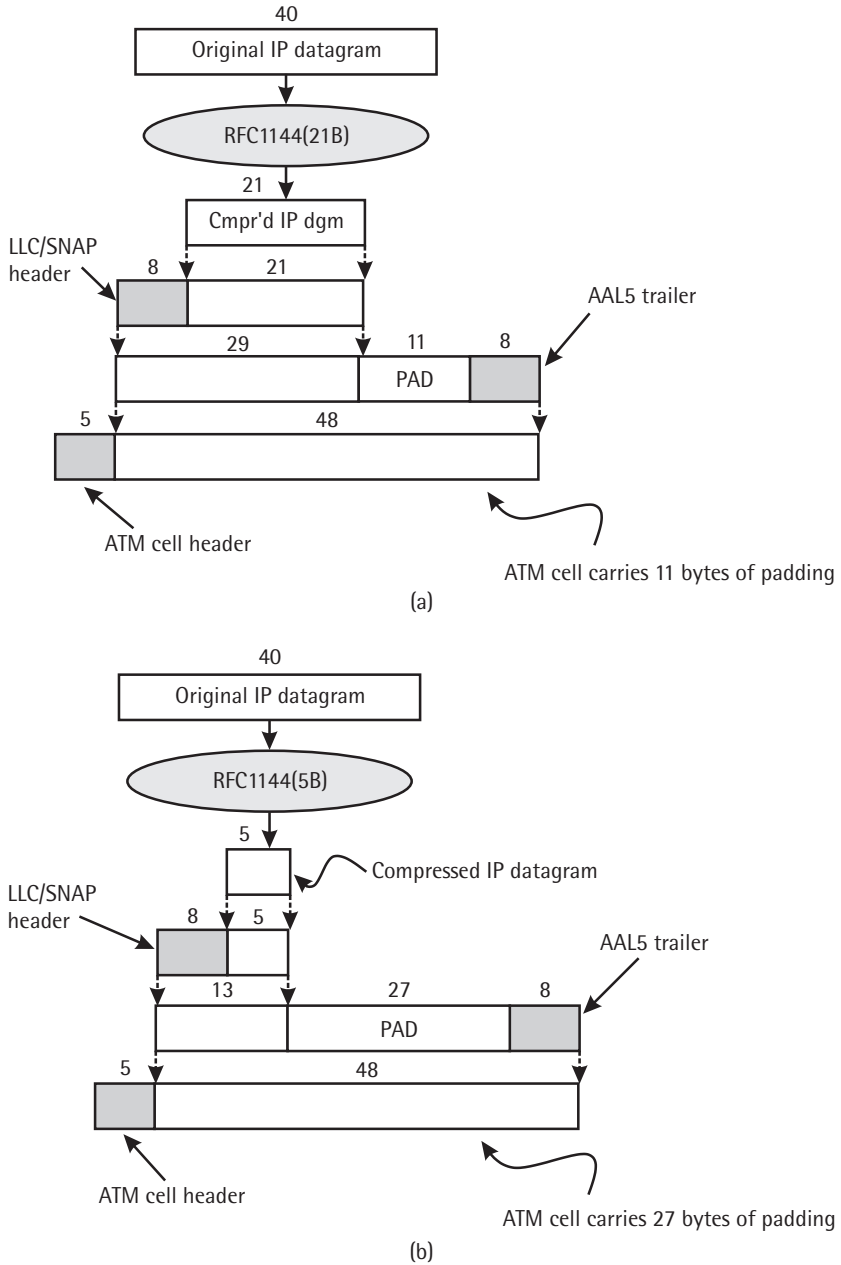


Figure 6.6 ATM encapsulation of a 40-byte IP datagram using RFC 1144(21B) and RFC 1144(5B) TCP/IP header compression.

6.4 Overall tax savings

As mentioned in Section 6.3, compression-related tax savings is more pronounced with small IP datagrams than with large datagrams. The overall tax savings derived from any given compression algorithm, therefore, depends on the distribution of IP datagram sizes in a given traffic sample. In this section, we compute the ATM cell tax for a sample Internet traffic trace using both VC-based multiplexing and RFC 1144, in isolation and in tandem. The traffic trace we consider is collected by the National Laboratory of Applied Network Research (NLANR) at FIX-West [3], the Federal InterEchange point in the United States.

The FIX-West sample [3] was recorded for 15 minutes on February 10, 1996, from 8:30 a.m. The total number of IP packets in the FIX-West sample is 11.7 million. Table 6.1 shows the top 10 dominant datagram sizes with respect to their frequency of occurrence; these datagrams comprise more than 70% of the total number of datagrams in the trace.

A quick glance through Table 6.1 reveals that the most common datagram length is 40 bytes, which occupies almost one-third of all datagrams. This results from ACK packets sent by the receiver and SYN or FIN packets when TCP connections are established or terminated. Next, IP datagram

Table 6.1
Top 10 Most Frequently Occurring IP Datagram Sizes From FIX-West Trace on February 10, 1996

IP Datagram Size (in Bytes)	Percentage	Rank
40	30.55	1
41	1.51	8
44	3.45	5
56	0.94	10
72	4.10	3
185	2.72	6
296	1.48	9
552	22.29	2
576	3.59	4
1,500	1.51	7
	Total: 72.14	

lengths of 552 and 576 bytes are due to the TCP MSS setting for different internetworking topologies. Following that, 1,500 bytes are due to the Ethernet MTU for end hosts connected to Ethernet LANs. IP datagrams of 41–44 bytes are mainly due to TELNET keystrokes. Other sizes are the result of IP fragmentation. The IP trace from the commercial MCI Internet backbone [4] reveals similar results in the order of percentages of IP datagram sizes but with different weights of percentages from FIX-West.

Table 6.2 shows the number of ATM cells needed to carry the top 10 most frequently found IP datagram sizes for the following encapsulation and compression techniques:

- No compression: LLC/SNAP encapsulation without any compression (typical encapsulation technique used by most ISPs);
- VC multiplexing;
- RFC 1144(5B);
- VC Mux+RFC 1144(5B): RFC 1144(5B) with VC multiplexing;

Table 6.2
Number of ATM Cells Needed to Carry Different Sizes of IP Datagrams for LLC Encapsulation, VC Multiplexing, RFC 1144 Compression, and VC Multiplexing and RFC 1144 in Tandem

	Number of ATM Cells					
IP Datagram Size (in bytes)	No Comp'n	VC Mux	RFC 1144(5B)	VC Mux + RFC 1144(5B)	RFC 1144(21B)	VC Mux + RFC 1144(21B)
40	2	1	1	1	1	1
41	2	2	1	1	1	1
44	2	2	1	1	1	1
56	2	2	1	1	2	1
72	2	2	2	1	2	2
185	5	5	4	4	4	4
296	7	7	6	6	7	6
552	12	12	12	11	12	12
576	13	13	12	12	12	12
1,500	32	32	31	31	32	32

- RFC 1144(21B);
- VC Mux+RFC 1144(21B): RFC 1144(21B) with VC multiplexing.

From Tables 6.1 and 6.2, we can calculate, for each encapsulation method, the total number of ATM bytes that need to be transmitted to carry the total number of IP bytes contained in the datagrams of sizes in the top frequency list in the FIX-West sample (which totals 11.7 million IP datagrams) as

$$\text{Total IP bytes} = \sum_{i=1}^{10} f_i \times 11{,}700{,}000 \times L_i \tag{6.1}$$

$$\text{Total ATM bytes} = \sum_{i=1}^{10} f_i \times 11{,}700{,}000 \times N_i \times 53 \tag{6.2}$$

where f_i is the percentage of the *ith* datagram size, L_i is the length (in bytes) of the *i*th datagram size, and N_i is the number of ATM cells needed to carry the *i*th datagram size. Here we assume that all datagrams carry TCP packets, as RFC 1144 is not valid for UDP/IP headers. Since UDP datagrams only constitute a small fraction of total Internet traffic, this assumption will still give us rough figures for the ATM cell tax for encapsulation methods involving RFC 1144.

Once the total ATM and IP bytes are obtained, the overall ATM cell tax for a given encapsulation method can be derived as

$$\text{Overall tax} = \frac{(\text{Total ATM bytes} - \text{Total IP bytes}) \times 100}{\text{Total ATM bytes}}\% \tag{6.3}$$

The overall ATM cell taxes for different compression methods are shown in Figure 6.7. The calculation of the overall ATM tax is based on the datagrams of sizes in the top 10 frequency list (Table 6.1). Since these datagrams comprise more than 70% of the total number of datagrams in the FIX-West sample, this calculation is a good measure of the overall ATM tax for this sample.

A few observations are in order. A hefty cell tax of nearly 21% is incurred for the typical LLC/SNAP encapsulation method without using any compressions. This result is consistent with the one reported for the

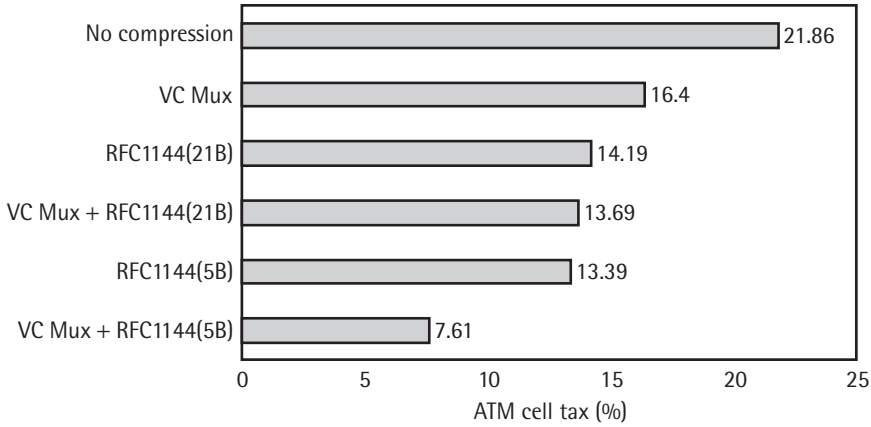


Figure 6.7 ATM cell tax for different encapsulation and compression techniques.

MCI traffic sample in [4]. VC multiplexing alone reduces the cell tax to 16.4%, and RFC 1144(21B) alone can reduce the tax to 14.19%.

Interestingly RFC 1144(5B) does not achieve any significant tax savings over RFC 1144(21B) when used with LLC/SNAP encapsulation. Therefore, for LLC/SNAP encapsulation, there is little motivation for using RFC 1144(5B) over RFC 1144(21B). Conversely, RFC 1144(5B) provides phenomenal tax savings over RFC 1144(21) when used with VC multiplexing; it brings the tax to a mere 7.61%.

6.5 Cells in frames

The primary source of the ATM cell tax is the five-byte ATM cell header for each 48-byte ATM cell payload. The cell header contains VC-specific information used by ATM switches to multiplex cells from different VCs. One interesting observation reveals that although headers of ATM cells from different VCs carry different information, headers of some consecutive ATM cells from the same VC do not change. Therefore, if a number of consecutive cells (with identical headers) from a given VC could be transmitted in one frame without interleaving cells from other VCs in the same frame, it would be possible to send only one cell header and suppress the

rest of the headers in the frame. At the destination, the headers of the cell payloads could be reconstructed by copying the only header in the frame.

For IP over ATM using AAL5, one AAL5 packet is carried over one or more ATM cells, which are transmitted over the same VC. Therefore, cells from the same AAL5 can be grouped into frames where only the first ATM cell carries a header, and the rest of the cells in the group carry only the 48-byte cell payload. ATM cells thus travel in variable size frames.

Xunet 2 [5], an early experimental wide-area ATM network, implemented a nonstandard framing format to carry a variable number of ATM cells with a single ATM cell header on DS3 trunks. For IP over ATM, the frame might contain one IP packet.

In 1996, a large number of industries and ATM vendors—including IBM, Bell Atlantic, Fore Systems, and Microsoft—formed the Cells in Frames (CIF) Alliance to promote an ATM-over-Ethernet technique called CIF. The CIF Alliance has specified a protocol [6] that allows ATM to be embedded into various frame-based legacy protocols (Ethernet and token ring) , using only one ATM header for up to 31 cells from the same VC. CIF was discussed in the ATM Forum [7].

Although the primary motivation behind CIF was to bring ATM to desktop without replacing Ethernet NICs with ATM NICs, one of the major benefits of CIF is the ATM tax cut due to cell header suppression. Figure 6.8 illustrates how an AAL5 packet is segmented into ATM payloads and encapsulated into an Ethernet frame using CIF. A CIF frame has eight bytes of header, including five bytes of ATM header (the same header for all cell payloads in the frame) and three bytes of CIF control information.

For up to 31 cell payloads, one AAL5 can be carried in one CIF/Ethernet frame. AAL5 packets longer than 31 cells will need to be carried in more

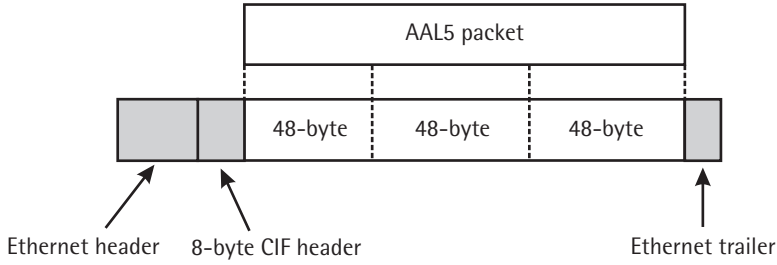


Figure 6.8 ATM cells in Ethernet frames.

than one CIF/Ethernet frame. Therefore, the number of frames needed to carry an IP datagram depends on the size of the datagram. Next, we show the computation of the ATM cell tax for CIF implemented in the FIX-West traffic trace.

The number of ATM payloads transmitted for an IP datagram of a given size using CIF depends on the type of encapsulation used at the adaptation layer. The second and third columns of Table 6.2 list the number of ATM payloads for the 10 most frequently found datagram sizes in the FIX-West trace for LLC/SNAP encapsulation and VC-based multiplexing, respectively.

As before, (6.1) can be used to compute the total number of IP bytes in the trace. Equation (6.2), however, is not valid for CIF, because 53-byte ATM cells are not transmitted in CIF. The correct way to compute the number of ATM bytes, including CIF overhead bytes, for an N-byte IP datagram is

$$\text{ATM bytes}\,(N) = \left\lfloor \tfrac{N}{31} \right\rfloor \times [(31 \times 48) + 8] + [(N \text{ modulus } 31) \times 48 + 8] \tag{6.4}$$

where (N modulus 31) means the remainder obtained from the division N/31. Equation (6.4) is based on the default operation of CIF where one CIF frame never carries more than one AAL5 packet. Using (6.4), we can compute the total number of ATM bytes, including CIF overhead bytes, for carrying all IP datagrams in the FIX-West trace as

$$\text{Total ATM bytes}_{\text{CIF}} = \sum_{i\text{-}1}^{10} f_i \times 11{,}700{,}000 \times \text{ATM bytes}\,(N_i) \tag{6.5}$$

where f_i is the percentage of the ith datagram size, and N_i is the number of ATM payloads to carry the ith datagram size. Once the total ATM bytes are computed for carrying total IP bytes, (6.3) can be used to compute the cell tax for CIF. Table 6.3 compares the cell taxes for CIF with the ones incurred when no CIF or other form of compression is used. The comparison in Table 6.3 shows that for the traffic trace of FIX-West, CIF decreases the cell tax from 22% to 16% when LLC/SNAP encapsulation is used and from 16% to merely 10% in a VC multiplexing environment.

Table 6.3
ATM Cell Tax With and Without CIF

Encapsulation Technique	NonCIF Tax	CIF Tax
LLC/SNAP	21.86%	15.93%
VC multiplexing	16.4%	10.21%

6.6 Practical use of cell tax reduction techniques

Figure 6.7 and Table 6.3 show the theoretical cell tax savings for a given IP traffic trace (FIX-West sample). The question that naturally arises is: Are these techniques being implemented by ISPs and carriers? In this section, we examine the practical limitations or difficulties ISPs or carriers face in implementing these techniques.

6.6.1 VC-based multiplexing

With VC-based multiplexing, each VC is associated with a given protocol. In an integrated environment with many different protocols such as IP and IPX, VC multiplexing will require separate VCs for separate protocols. Large corporations usually use ATM PVCs to provide connectivity between two sites. If the PVC is configured to take advantage of VC-based multiplexing, once a PVC is set up, and a particular protocol associated with it, the PVC cannot be used to transmit data packets carrying another protocol. If multiple PVCs are set up for multiple protocols, certain PVCs may remain underutilized while others may remain overloaded at certain times, as the traffic of one PVC cannot be transmitted over another PVC. Maintaining multiple PVCs to the same destination, therefore, may prove very costly. Due to these reasons, ISPs and carriers continue to use LLC/SNAP encapsulation, which allows them to reuse the same PVC for all protocols.

In an IP-only environment, the multiplexing of different protocols is not relevant and hence VC-based multiplexing should be a preferred method of encapsulation over LLC/SNAP encapsulation. Moreover, as ATM SVCs are likely to replace PVCs in the future, it will be much easier to implement VC-based multiplexing even in a multiprotocol environment, because a new VC can be set up automatically whenever needed using signaling and closed down when not needed.

6.6.2 TCP/IP header compression (RFC 1144)

TCP/IP header compression has been implemented in BSD kernel and SLIP software distribution and used by hosts connected to low-speed serial links for several years. There are two main reasons why TCP/IP compression is not widely used for ATM links. The first one is the increased processing overhead due to compression and decompression of headers. While this overhead is not relevant for slow-speed modem connections, it becomes a bottleneck for high-speed ATM links where the number of TCP/IP packets to process per second is very large. A recent implementation [8] of TCP/IP header compression over a 155-Mbps ATM link in a DEC Alpha platform revealed that, instead of increasing, TCP throughput actually degrades when TCP/IP header compression is implemented. This is because the extra compression overhead slows down the CPU and prevents it from achieving the maximum throughput.

The DEC Alpha implementation [8], however, showed that TCP throughput did increase for 2-Mbps links. Since wireless ATM links are 2 Mbps, and bandwidth efficiency is of prime concern for such wireless links, it is expected that TCP/IP header compression will be adopted in ATM wireless links. Given the steady increase in processor speed over the last few years, TCP/IP header compression may also be adopted for high-speed ATM links in the future when high-speed processors become a commodity. An alternative to using high-speed processors would be to develop and market special-purpose hardware for TCP/IP header compression for wide acceptance of TCP/IP header compression as a measure for reducing the ATM cell tax for high-speed trunks in the near future.

The second reason why TCP/IP header compression is not popular with ATM networks is that separate VCs need to be open for each TCP connection for the compression to work. This poses a scalability problem for high-capacity trunks carrying thousands of TCP connections. If the cost of opening and maintaining VCs falls sharply in the future, TCP/IP header compression may become an attractive solution to the cell tax problem.

6.6.3 Cells in frames

CIF products are beginning to emerge among ISPs and carriers. However, CIF is a LAN technology and hence saves cell tax on LAN links. Accordingly, the potential of CIF in reducing ATM cell tax for WAN links has

caught the attention of long-haul carriers, and there is work in progress to support CIF over SONET and PPP to support long-distance ATM links.

6.7 Summary

VC multiplexing and TCP/IP header compression are two standard methods for ATM tax savings. For typical Internet traffic, it is possible to save up to 15% cell tax using these methods. With these techniques, there are no changes required at the ATM layer. However, opening and maintaining a large number of VCs remains the main hindrance to wide adoption of these techniques by ATM network operators.

While VC multiplexing and TCP/IP header compression are implemented above the AAL, CIF—a LAN-based cell-tax–saving technique—works below the AAL. CIF reduces the cell tax by carrying only one cell header for a group of ATM cells with the same header. CIF can reduce cell tax by 6% for traffic patterns commonly found on the Internet backbones. Work is in progress to develop CIF for SONET links that will allow long-haul carriers to benefit from this technology in reducing the ATM cell tax on expensive long-distance links.

References

[1] Heinanen, J., "Multiprotocol Encapsulation Over ATM Adaptation Layer 5," *Internet RFC 1483*, IETF, July 1993.

[2] Jacobson, V., "Compressing TCP/IP Headers for Low-Speed Serial Links," *Internet RFC 1144*, IETF, February 1990.

[3] FIX-West Web site: http://www.nlanr.net/NA/Learn/packetsizes.html. Accessed on May 3, 1999.

[4] Thomson, K., G. J. Miller, and R. Wilder, "Wide Area Internet Traffic Patterns and Characteristics," *IEEE Network*, Vol. 11, No. 6, December 1997, pp. 10–23.

[5] Fraser, A. G., et al., "Xunet 2: A Nationwide Testbed in High-Speed Networking," *Proc. IEEE INFOCOM*, May 1992, pp. 582–589.

[6] CIF Alliance, "Cells in Frames Version 1.0: Specification, Analysis, and Discussion," 31 July, 1996. Available from: http://www.ziplink.net/Iroberts/Atmf-961104.html.

[7] Roberts, L. G., "Request for Coordination of Cells in Frames Specification," ATM Forum contribution 96-1104, August 19, 1996.

[8] Buchanan, B., "TCP/IP Header Compression Over ATM," Masters Thesis, Department of Electrical Engineering and Computer Science, University of Iowa, Iowa City, IA, USA, 1994.

7

Improving TCP Performance Against ATM Cell Loss

7.1 Introduction

ATM is based on statistical multiplexing. This implies that the network allocates network resources based on the assumption that all the sources will not need their peak resources at the same time. Depending on the level of multiplexing and the burstiness of the sources, an ATM network can encounter congestion during normal operation. Because ATM is based on a lightweight protocol, there is no mechanism for the retransmission of cells lost due to congestion. An ATM network, therefore, attempts to minimize cell loss. This chapter discusses measures that can be used to increase the throughput of TCP by reducing cell loss in ATM networks.

There are a number of solutions to increase the throughput of ATM networks. This book groups these solutions into four categories as shown in Figure 7.1. Although all the solutions use buffers in the switches to temporarily store cells during periods of network congestion, solutions in the first category rely only on buffering to fight against cell loss. For this category, the chapter describes various types of buffering in ATM switches, depending on the location of the buffers. In addition, the chapter describes

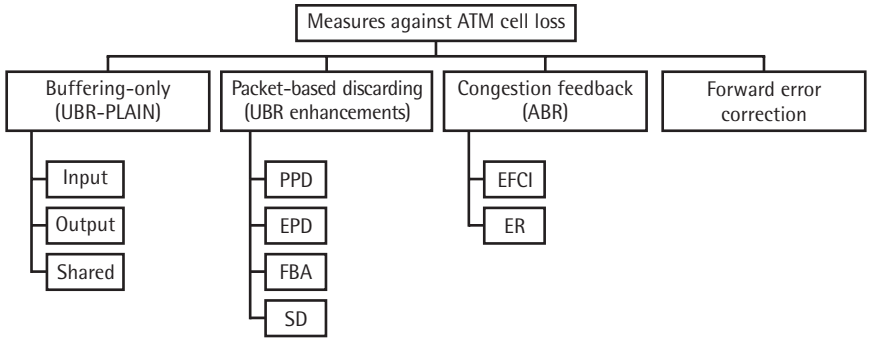


Figure 7.1 Four categories of solutions to increase the throughput of ATM networks.

their relative advantages and disadvantages in terms of buffer size requirement, ease of buffer management, and implementation complexity.

The solutions under the second category shown in Figure 7.1 use packet-based discarding strategies at the switches for traffic coming from packet-based applications (e.g., TCP). The techniques are based on concentrating cell losses in fewer packets to reduce the packet-corruption rate. This chapter discusses two such techniques, partial packet discard (PPD) and early packet discard (EPD).

The third category of solutions listed in Figure 7.1 uses feedback from the network to the sources to inform the sources to slow down during periods of network congestion. The sources can ramp up their traffic rates later after the congestion alleviates.

Techniques relying on forward error correction (FEC) to recover from lost cells fall under the fourth category listed in Figure 7.1. If a cell is corrupted by errors in an ATM network, FEC can be used to reconstruct the cell at the destination, thereby reducing the effect of cell loss in the network and increasing the throughput of the network.

7.2 Buffering in ATM switches

ATM is based on statistical multiplexing. An ATM switch consists of switching fabric that is used to route cells through the switch. Small switches could use a high-speed bus as the fabric, whereas larger switches

typically use multiple stages of small crossbar switching elements (SEs) to carry out the switching function. A switch can be classified as nonblocking or blocking. In a blocking type of switch, a cell may be blocked from being routed to the output due to internal routing conflicts. Buffers are used to temporarily store cells that lose the conflict. A nonblocking type of switch does not suffer from internal routing conflicts; a cell can only be blocked from routing in the case of output port contention (i.e., when two or more cells are destined to the same output).

During times of congestion, the amount of data arriving at an ATM switch may exceed the data throughput rate of the outgoing lines. Buffers are used in the switches to avoid cell losses in such cases. Buffers can be located at the inputs or outputs of the switch, or they can be shared between the inputs and outputs of the switch.

The fundamental difference between the locations of the buffers is in the amount of buffering required, the speed of the buffers, and the complexity of managing the buffer as described in Sections 7.2.1–7.2.4. The proper placement and arrangement of buffers have a dramatic impact on the performance of the switches. Sections 7.2.1–7.2.4 describe the location of buffers in a switch and their impact on the performance of the switch.

7.2.1 Input-buffered switches

The buffers in an input-buffered switch are located at the input ports of the switch as shown in Figure 7.2. If a cell arriving on an input port during a cycle cannot leave the switch during the same cycle due to output port contention (two or more cells at different input ports destined to the same output port during a cycle is called *output port contention*), the cell is stored in the input buffer and is routed during a subsequent cycle.

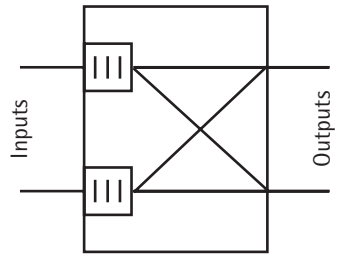


Figure 7.2 Input-buffered switch.

The input buffers are managed as simple FIFO queues. A buffer, therefore, needs to operate at the same speed as the line speed of the input port. Internal speedup of the switch is not required, and the hardware can be less complex than that of other buffering schemes as discussed in Sections 7.2.2–7.2.4. However, when a cell at the head of a queue waits for transmission to its destined output port, successive packets (which may be destined to different output ports) in the queue must also wait. This phenomenon, called head-of-line (HOL) blocking, reduces the throughput of input-buffered switches as illustrated by the following example.

Let's say that during a cycle, two cells, destined for output ports O_1 and O_2, are queued at the input buffer at I_1, and a cell destined to O_2 is queued at the buffer at I_2 as shown in Figure 7.3. Therefore, there are two cells at the head of two buffers that are destined to the same output (i.e., O_2). Let's say that the switch selects the cell from the buffer I_2 to go to O_2 during this cycle. In this case, the head of queue cell from input I_1 cannot leave during that cycle, and it also blocks the second cell in the queue, which is destined to O_1. Although output port O_1 is not used during this cycle, the second cell from input I_1 still cannot leave the queue since it is blocked by the HOL cell at input I_1. As a result, the bandwidth of the line connected to output port O_1 is wasted during this clock cycle, resulting in a low throughput of input-buffered switches.

Allowing cells behind the head of the queue to leave the buffer when the head of the queue cell is blocked can overcome the throughput reduction due to HOL blocking. This technique, known as *window bypass*, consists of searching for cells in a fixed window (behind the head of queue cell) to see if any can be routed to the output while the head of the queue cell is blocked.

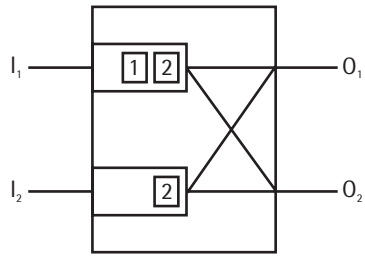


Figure 7.3 Illustration of HOL blocking in input-buffered switches.

7.2.2 Output-buffered switches

Output-buffered switches use buffers at each output of the switch as shown in Figure 7.4. A buffer must be able to receive up to N packets during a clock cycle, where $N \times N$ is the size of the switch. Output-buffered switches do not suffer from the HOL blocking effect as in the case of input-buffered switches and hence have higher throughput than input-buffered switches.

The main drawback of output-buffered switches is that they need to operate N times faster than an input line of the switch. This is to ensure that no cells are lost when all the inputs have cells destined to the same output. This higher speed increases the implementation complexity and cost of the switch. Switches can also be constructed by using both input and output buffers. In this case, the operating speed of the switch can be lower than in the case of the output-buffered switch [1]. Dedicated buffers for the outputs may result in low buffer utilization resulting in large buffer requirement to achieve a given cell loss.

7.2.3 Shared-buffered switches

The buffers in a shared-buffer switch are shared between all the inputs and outputs of the switch as shown in Figure 7.5. In a shared-buffer switch of size $N \times N$, all inputs and outputs have access to the shared buffer, which should be capable of writing up to N incoming and reading up to N outgoing cells in the same clock cycle. There is no HOL blocking in shared-buffer switches, and optimal throughput and delay performance is achieved. Furthermore, buffer utilization is better in shared-buffer switches than it is in input- or output-buffered switches. Hence, the buffer size can be smaller than it is in input- or output-buffered switches for the same cell loss performance.

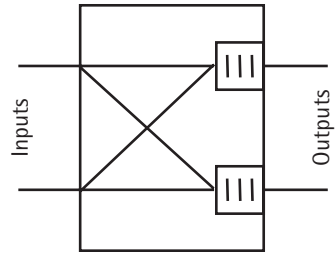


Figure 7.4 Output-buffered switch.

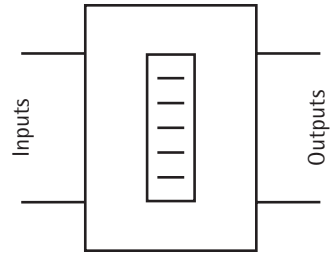


Figure 7.5 Shared-buffered switch.

A shared-buffer switch can easily be modified to handle several service classes through priority control functions. Multicasting and broadcasting can also be easily implemented. The limitations of shared-buffer multistage interconnection networks (MINs) arise from technological limitations. A buffer in a switch needs to queue N incoming and dequeue N outgoing cells per clock cycle. Therefore, the bandwidth of a buffer must be at least the sum of the bandwidths of the incoming and the outgoing lines.

7.2.4 Comparison of buffering strategies

A number of architectures can be used in the switching fabric of an ATM switch. Such architectures include crossbar, multiple bus, and multistage switches built using MINs. Because of cost effectiveness, multistage switches are used in large commercial switches. A multistage switch consists of a number of stages of small crossbar SEs, as shown in Figure 7.6. The buffering in the switch is distributed among all the SEs. The SEs can use input, output, or shared buffering as described in Sections 7.2.1–7.2.3.

The performance of multistage switches can suffer from output port contention when a number of cells are destined to the same output. In a blocking type of multistage switch, the performance also depends on internal routing conflicts between cells arising due to the interconnection pattern used to connect the SEs. This section compares the throughput and delay of input-, output-, crosspoint-, and shared-buffered switches in the case when the distribution of traffic among the outputs is uniform [2].

Figures 7.7 and 7.8 show the throughput and delay as a function of the *offered traffic load* (defined as the probability of a cell arriving at an input during a cycle) at the inputs of 64×64 switches using buffer sizes of six cells per port. The switches used the blocking delta multistage interconnection

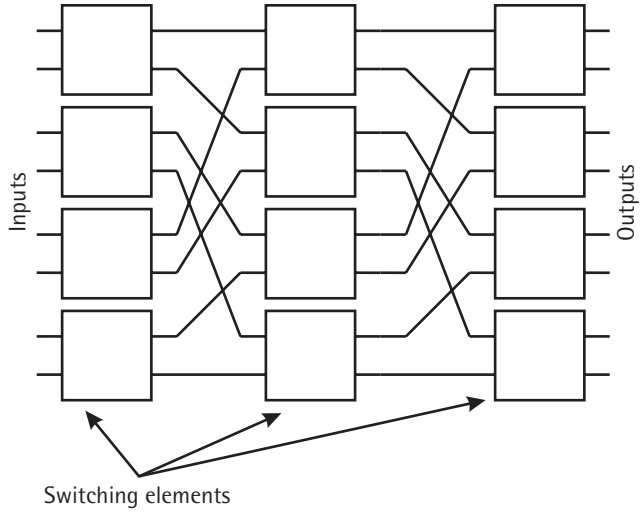


Figure 7.6 A multistage switching fabric.

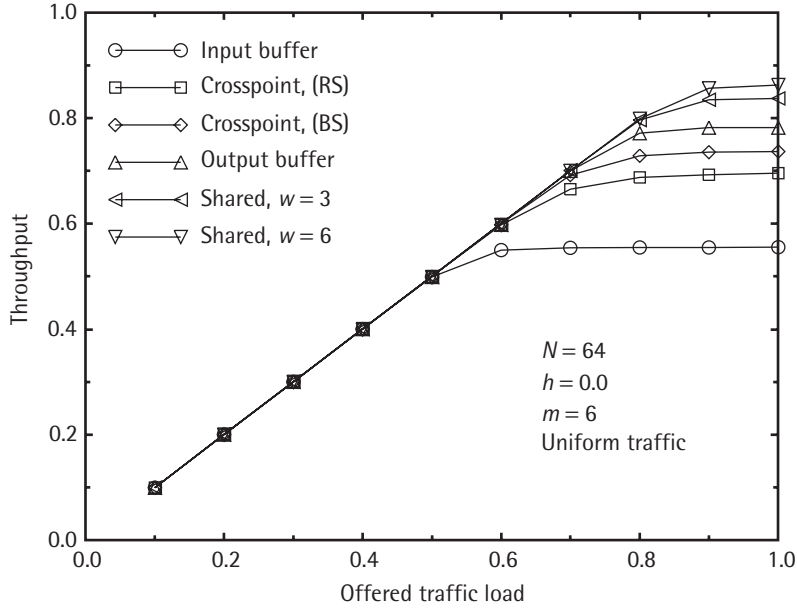


Figure 7.7 Switch throughput as a function of the offered load to the switch.

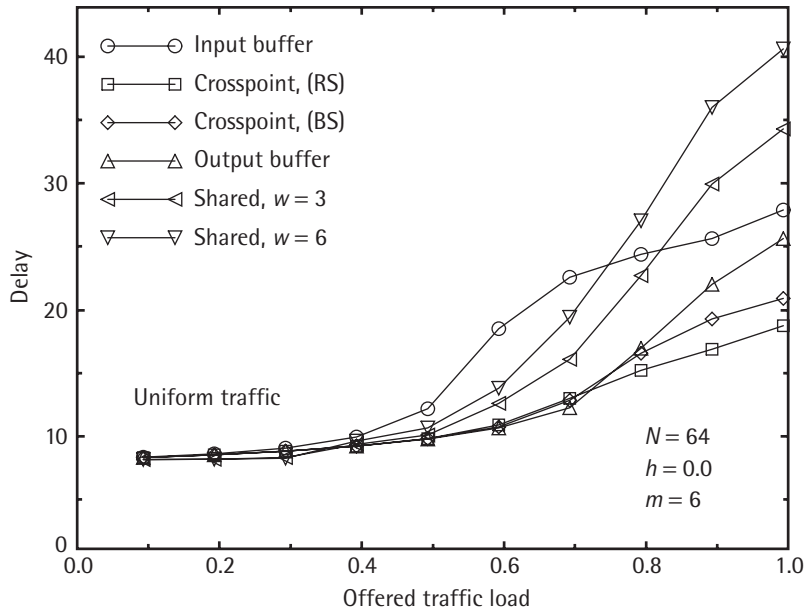


Figure 7.8 Delay of cells through the switch as a function of the offered load to the switch.

between the stages. Internal routing conflicts between the cells were randomly resolved. Because there is not much internal routing conflict inside the switch at low-offered traffic loads, the throughputs of the different buffering schemes are similar.

At high-offered traffic loads, the performance of the shared buffer is the highest because multiple cells can be queued and dequeued during the same cycle. The performance of the input buffer is the lowest because of HOL blocking. For the shared-buffer switch, where the buffer is organized as a number of FIFO logical queues, we have used two different bypass window sizes ($w = 3$ and $w = 6$) for removing the effects of HOL blocking. For the crosspoint buffer, we have used two different conflict resolution techniques:

- Random selection (RS), where the internal routing conflicts are resolved randomly (i.e., a cell is randomly selected from the cells in conflict).

- Blocked selection (BS), where a blocked cell is given priority over a new cell in the case of routing conflicts.

Figure 7.8 shows the delay of cells through the switch as a function of the offered traffic load, assuming that cells spend at least one clock cycle in each stage. It is found that because of no congestion, the delay is small when the offered load at the input is small. However, an increase in the offered traffic load causes congestion in the buffer, resulting in a sharp increase in the delay. The delay is the least for crosspoint buffers and is the maximum for shared buffers. This is because the shared buffers store much more conflicting cells and therefore lose less cells, which, in turn, results in higher queue delays.

Figures 7.9 and 7.10 compare the throughput and delay of the various buffering schemes as a function of the buffer size for a 64×64 switch and a maximum offered traffic load ($r = 1$). For the crosspoint buffer, RS and BS have been used to resolve routing conflicts inside the switch. Initially, when the buffer size is increased, the throughput increases because of lower cell

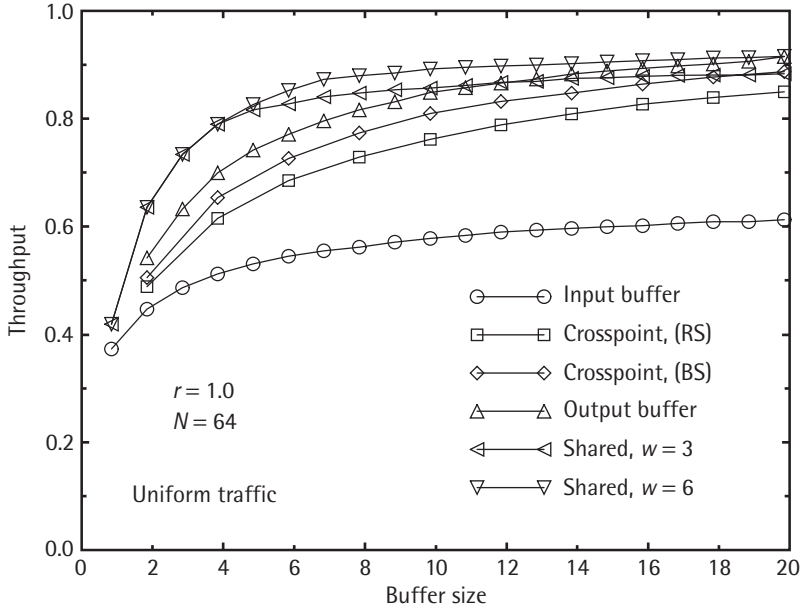


Figure 7.9 Throughput of the switch as a function of buffer size.

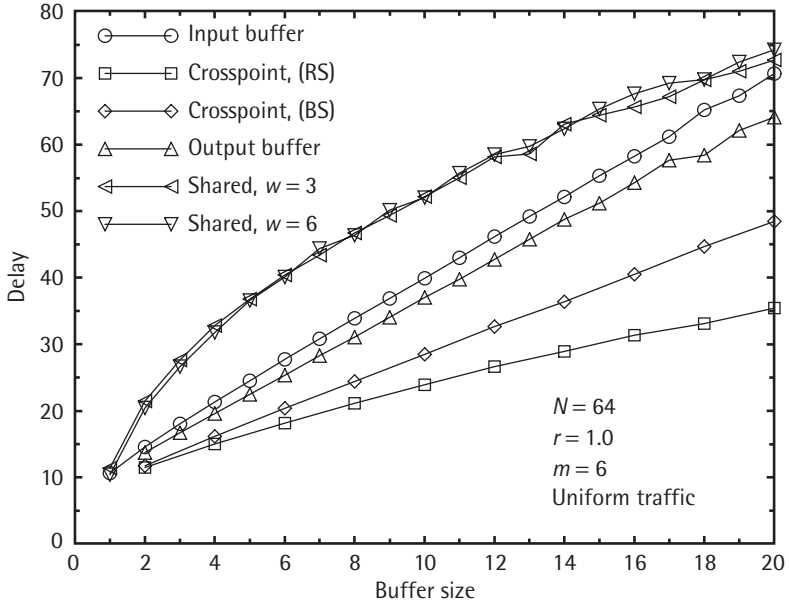


Figure 7.10 Cell delay through the switch as a function of buffer size.

losses from larger buffer sizes. However, if the buffer size continues to increase, the buffers become congested because of the inability of the output lines to carry traffic. At this point, an increase in the buffer size does not help to increase the throughput. The throughput is no more a function of the buffer size, and, according to Little's Theorem, the delay increases as the buffer size increases. The throughput of the shared buffer is the best, while that of the input buffer is the lowest because of HOL blocking.

For small buffer sizes in the switches, the throughput is lower because of cell losses due to insufficient buffer space to hold the incoming cells. However, when the buffer space is increased, the cell losses are predominantly the result of internal routing conflicts and buffer saturation. Consequently, the throughput does not increase with an increase in the buffer space.

7.3 Packet-based discarding

TCP is a reliable end-to-end protocol. TCP sends variably-sized packets to the destination and waits for acknowledgments to arrive from the

destination. If the sender does not receive an acknowledgment for a packet within a certain amount of time, it retransmits the packet.

ATM is being widely used to interconnect LANs based on the TCP/IP protocol. Because ATM is based on small packets called cells, an AAL breaks a TCP packet into a number of small ATM cells so that they can be transported over the ATM network. The cells are reassembled at the destination to reconstruct the TCP packets.

ATM is a lightweight protocol. It was originally designed to operate with high-speed, error-free links. To reduce the processing load at the ATM switches, the protocol does not do the following.

- Perform any error checking of the cell payload. ATM relies on the upper layer at the destination to carry out any error checking of the payload. ATM, however, does perform error checking of the ATM cell header (containing the destination address) to prevent any misrouting of cells.
- Retransmit cells that are lost due to network congestion.

As explained in Section 5.3, the second item, retransmitting cells, has a detrimental effect on the performance of TCP. If an ATM switch drops a cell, the destination TCP process flags the packet (to which the dropped cell belongs) as a corrupted packet, drops the entire packet, and refrains from sending an acknowledgment. In the absence of an acknowledgment, the source TCP retransmits the entire packet. Although the wireline ATM network can be assumed to have a very low BER, cell loss due to congestion in the network is more frequent. The combined effect of cell loss and lack of error correction in an ATM network results in retransmission of packets from the source and significantly affects the performance of TCP, which is a reliable connection-oriented protocol to the application layer.

Since the plain UBR service does not have any additional mechanism (other than having buffers) to address the cell loss problem, TCP performs poorly over UBR. From now on, we will refer to the plain UBR service as UBR-PLAIN. To improve TCP performance over the UBR-PLAIN, several techniques have been proposed. Sections 7.3.1–7.3.5 describe these techniques.

7.3.1 Partial packet discard (PPD)

The loss of a single cell of a TCP packet renders the packet useless. Therefore, dropping all cells of a corrupted packet can help the network to

operate more efficiently. When a cell belonging to a packet is dropped by an ATM switch, the remaining cells of the packet are useless (to the destination) and hence should be dropped by the switch rather than be carried over the network, under the PPD technique. PPD increases the bandwidth utilization of the network and decreases the delay of the good packets (i.e., packets that can reach the destination without any errors). We will refer to UBR service augmented with PPD as UBR-PPD.

7.3.2 Early packet discard (EPD)

EPD is a second scheme that has been proposed to increase the efficiency of TCP over UBR-PLAIN [3]. In this scheme, a buffer in a switch has a threshold. A buffer reaching the threshold indicates the onset of congestion, which may lead to cell loss. Since partial TCP packets are of no use at the destination, once the threshold has been reached, the EPD scheme admits cells (if there is buffer space) from only those packets that have already entered the switch; cells belonging to packets that have not yet entered the switch are dropped.

A switch implementing EPD drops cells belonging to fewer packets (which then must be retransmitted) instead of randomly dropping cells. If the switch dropped cells randomly without considering packets, a larger number of packets would have been affected (resulting in larger number of retransmissions), although the number of cells lost in both the cases could have been similar. In essence, the EPD scheme spreads the loss of cells over fewer packets resulting in fewer retransmissions from the source TCP. UBR service augmented with EPD cell dropping algorithm will be referred to as UBR-EPD.

Although PPD and EPD are both packet-based cell-dropping algorithms, it is interesting to compare their performance. PPD discards partial packets in reaction to the loss of a cell belonging to a packet. PPD is, therefore, a reactive scheme. EPD, on the other hand, starts discarding packets even before a cell has been lost due to congestion (i.e., it discards cells in anticipation of impending congestion). EPD is, therefore, a proactive packet-based congestion-control mechanism.

Simulation studies to enable comparison of the performance of TCP over UBR-PLAIN, packet TCP, UBR-PPD, and UBR-EPD have been carried out by Romanow and Floyd [3]. The simulation consisted of 10 TCP sources. In TCP over UBR-PLAIN, the ATM switch drops cells from the tail of the buffer when the buffer is full. This is sometimes referred to as

drop-tail packet discard. No ATM switch is used in packet TCP (i.e., the sources and the destination are connected directly using the TCP/IP protocol). In the case of TCP over UBR-PLAIN, PPD, and EPD, the 10 sources were connected to an ATM switch that was connected to a receiver using a high-speed link. Normalized *goodput*, defined as the fraction of the link bandwidth that carries useful cells, has been used as the performance measure.

Table 7.1 presents the results of the studies comparing TCP over UBR-PLAIN, packet TCP, UBR-PPD, and UBR-EPD [3]. Each packet consisted of 9,180 bytes, and the EPD threshold was set to half the ATM switch buffer size. A 64-Kbyte maximum TCP window size was used. From Table 7.1, we make the following observations.

- TCP performs very poorly over UBR-PLAIN for a limited buffer with significant cell loss. Increasing the buffer size increases TCP throughput. To achieve good TCP performance over UBR-PLAIN, large buffers are needed.
- UBR-PPD and UBR-EPD can achieve good TCP performance with small buffers.
- UBR-EPD performs slightly better than UBR-PPD and similarly to packet TCP.

7.3.3 EPD with fair buffer allocation

Despite its capability of significantly improving TCP throughput, EPD suffers from a fairness problem. When many ATM VCs carrying TCP traffic are multiplexed in an UBR-EPD switch, the available bandwidth is not

Table 7.1
TCP Throughput Over Various Flavors of UBR Service

Buffer Size (in Kbytes)	UBR-PLAIN	UBR-PPD	UBR-EPD	Packet TCP
100	0.62	0.8	0.98	0.98
200	0.7	0.88	0.98	0.98
300	0.8	0.92	0.98	0.98
400	0.84	0.95	0.98	0.98

equally shared by all TCP sources; some sources receive high bandwidth at the expense of others.

The reason for unfairness in EPD is due to the fact that when EPD is invoked, the switch tries to drop the next incoming full packet from any VC irrespective of its current load in the buffer. For connections with small packets, arriving packets will be dropped even though the VC is using less than its fair share of the buffer space. Consequently, connections with larger packets have a better chance of getting more throughput than connections with smaller packets. Simulation studies [4, 5] have confirmed the unfairness problem of UBR-EPD.

Fair buffer allocation (FBA) [5] is an enhancement of EPD to address the above-mentioned fairness problem in UBR-EPD switches. A good explanation of how FBA works can be found in [6]. We will refer to EPD enhancement with FBA as UBR-EPD-FBA.

FBA keeps per-VC accounting; it calculates *N(i),* the number of cells in the buffer for VC *i*. Figure 7.11 illustrates the parameters of FBA. EPD is invoked (FBA enters discard mode) when buffer occupancy reaches threshold *R*. *B* is the buffer size, and *N* is the number of cells in the buffer. When *N* is larger than *R*, FBA enters discard mode.

The central concept of FBA is to prevent a VC from snatching more than its fair share of available bandwidth. To implement this, FBA tries to discard packets from those VCs that are occupying more than their fair share of buffer space. If there are *V* active VCs, VCs with at least one cell in the buffer, the fair share of buffer is simply *N/V*. VC *i* is underutilizing its fair share for $N(i) < N/V$ and overutilizing for $N(i) > N/V$. The following weight is calculated to assess the current utilization

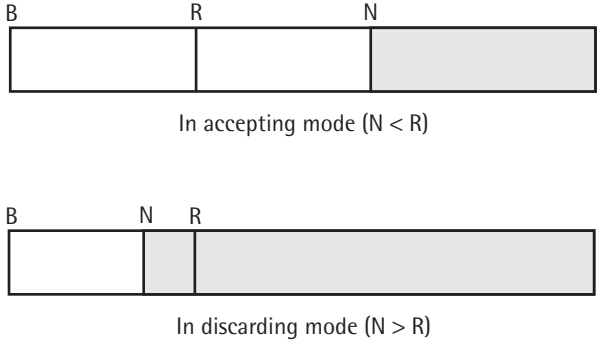


Figure 7.11 Buffer parameters for FBA.

$$W(i) = \frac{N(i)}{N/V}$$

Any *W(i)* greater than one means the VC is using more than its fair share. FBA causes a packet to be dropped from VC *i* if both of the following conditions are met:

$$(N > R) \tag{7.1a}$$

and

$$W(i) > Z \times \left(\frac{B-R}{N-R} \right) \tag{7.1b}$$

where the optimum value for Z was found to be little less than one [4]. From (7.1) the following observation is in order: FBA is a dynamic algorithm; it adjusts to the level of congestion experienced in the switch. As N increases (congestion increases), *W(i)* is compared to smaller values. As a result, packets from more VCs are dropped at a higher level of congestion. Section 7.3.5 discusses the performance of TCP over UBR-EPD-FBA.

7.3.4 EPD with selective drop (SD)

A simpler version of FBA, called selective drop (SD), was proposed by Goyal et al. [4]. Instead of being dynamic, SD uses the following fixed condition to determine whether to drop packets from VC *i*:

$$(N > R) \tag{7.2a}$$

and

$$W(i) > Z \tag{7.2b}$$

As we can see from (7.2), SD continues to compare *W(i)* to the fixed value Z independent of N. This means that with SD, relatively fewer VCs will be dropped during a high level of congestion (large N).

7.3.5 Comparison of UBR options

It is interesting to assess the benefits of FBA and SD—both based on per-VC accounting—over UBR-PLAIN and UBR-EPD. Goyal et al. [4]

conducted a thorough simulation study for both LAN and WAN configurations, reported TCP throughput and fairness results for FBA and SD, and compared them with UBR-PLAIN. The simulation study and the results from [4] are presented below.

The simulations were run with a network configuration of N identical, infinite TCP sources sending data to N TCP destinations via two common ATM switches implementing various UBR options. All TCP sources start transmission at the same time to maximize the probability of TCP synchronization (for worst case simulation). Simulations were run for both LAN and WAN distances with varying buffer sizes at the switches. Table 7.2 summarizes other important parameters used in the simulation.

Two performance measures, normalized TCP throughput and fairness, were collected after each simulation. Normalized TCP throughput is the ratio of the aggregate TCP throughput (sum of each source's throughput) achieved to the maximum throughput possible on a 155.52-Mbps SONET link. The fairness is measured as

$$\text{Fairness} = \frac{\left(\sum x_i\right)^2}{N \times \sum x_i^2}$$

Table 7.2
Simulation Parameters

Parameter	Value
Round-trip propagation delay for LAN	30 μs
Round-trip propagation delay for WAN	30 ms
All link bandwidths	155.52 Mbps
PCR	155.52 Mbps
TCP segment size	512 bytes
TCP timer granularity	100 ms
TCP maximum receiver window size for LAN	64K
TCP maximum receiver window size for WAN	600K
TCP ACK delay timer	Not set; segments are acknowledged as soon as they are received

where N is the number of TCP sources and x_i is the throughput of source i. The above fairness formula gives a normalized measure of the dispersion of the individual throughputs.

Tables 7.3 and 7.4 summarize the throughput and fairness results obtained from the simulation study. The following observations may be made from these results:

Table 7.3
TCP Throughput for Various UBR Options

Configuration	Number of Sources	Buffer Size (cells)	TCP Throughput (normalized)			
			UBR-plain	EPD	SD	FBA
LAN	5	1,000	0.21	0.49	0.75	0.88
LAN	5	3,000	0.47	0.72	0.90	0.92
LAN	15	1,000	0.22	0.55	0.76	0.91
LAN	15	3,000	0.47	0.91	0.94	0.95
WAN	5	12,000	0.86	0.90	0.90	0.95
WAN	5	36,000	0.91	0.81	0.81	0.81
WAN	15	12,000	0.96	0.92	0.94	0.95
WAN	15	36,000	0.92	0.96	0.96	0.95

Table 7.4
TCP Fairness for Various UBR Options

Configuration	Number of Sources	Buffer Size (cells)	TCP Fairness (normalized)			
			UBR-PLAIN	EPD	SD	FBA
LAN	5	1,000	0.68	0.57	0.99	0.98
LAN	5	3,000	0.97	0.84	0.99	0.97
LAN	15	1,000	0.31	0.56	0.76	0.97
LAN	15	3,000	0.80	0.78	0.94	0.93
WAN	5	12,000	0.75	0.94	0.95	0.94
WAN	5	36,000	0.86	1.00	1.00	1.00
WAN	15	12,000	0.67	0.93	0.91	0.97
WAN	15	36,000	0.77	0.91	0.89	0.97

- Both FBA and SD significantly improve TCP throughput and fairness over UBR-PLAIN for both LAN and WAN.
- Both FBA and SD significantly improve throughput and fairness over UBR-EPD for LAN; for WAN there is no significant difference.
- Although FBA is more complex than SD, it does not provide significantly better results than SD.

7.4 Feedback congestion control

UBR-PPD, UBR-EPD, and UBR-EPD-FBA are low-cost solutions to the cell loss problem. During network congestion, they focus on cell-discarding techniques without providing any congestion feedback to the sources. The focus is more on reducing the effect of cell loss on TCP performance (by minimizing the spread of cell loss among TCP/IP packets), rather than reducing the cell loss itself.

An alternative but more expensive solution to the cell loss problem is to provide explicit congestion feedback to sources. Cell loss can be drastically reduced if sources adjust their rates according to the network congestion level. ABR service implements such congestion feedback mechanisms using RM cells. ABR sources periodically probe the network congestion state with RM cells sent intermixed with data cells, as shown in Figure 7.12. The RM cells are turned around at the destination and sent back to the source. Along the way the switches may write congestion information in the RM cells. There are two types of congestion feedback, ABR-EFCI and ABR-ER.

7.4.1 ABR-EFCI

If a switch experiences or detects the onset of a congestion, it marks the explicit forward congestion notification (EFCI) bit in the header of a data

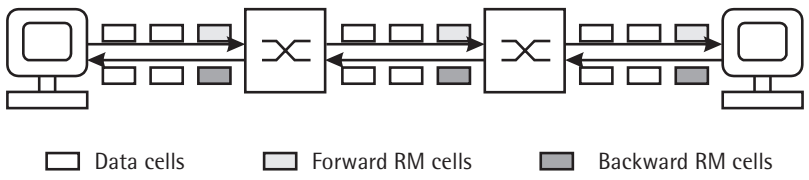


Figure 7.12 Flow of data and RM cells in an ABR connection.

cell. When the destination returns an RM cell to the source, it sets the congestion indication (CI) bit in the RM cell if the last data cell received had its EFCI bit set. Upon receiving an RM cell, the source either increases or decreases its rate depending on whether the CI bit is set or not. This is a binary feedback mechanism, whereby the source learns whether the network is congested or not; it cannot learn the degree of congestion.

7.4.2 ABR-ER

With explicit rate (ER) feedback, an ATM switch employs a control algorithm to monitor the current available bandwidth for the ABR service and the number of ABR VCs trying to share the bandwidth. The switch algorithm calculates the fair share for each VC and writes this fare share in the ER field of a backward RM cell (containing a higher value of ER). Upon receiving an RM cell, the source simply adjusts its current rate according to the ER value contained in the RM cell.

7.4.3 Performance of ABR-EFCI and ABR-ER

Performance of TCP over ABR depends on many ABR parameters including the PCR. It is, therefore, not appropriate to provide general quantitative results for TCP performance over the ABR service. This section presents some qualitative observations on the benefit of ABR-EFCI and ABR-ER over the UBR-PLAIN service based on the results reported by Saito et al [7].

Table 7.5 shows TCP throughput achieved over UBR-PLAIN and ABR-EFCI services when 25 VCs are multiplexed with a buffer size of 4,096 cells at the ATM switch. The first two rows of Table 7.5 show the results obtained with a PCR of only 25 Mbps (low-speed ATM connection).

Table 7.5
Performance of TCP Over ABR-EFCI for High and Low PCRs
(25 VCs, 4,096 Cell Buffers)

Service	PCR (Mbps)	Aggregate TCP Throughput (Mbps)
UBR-PLAIN	25	107.0
ABR-EFCI	25	127.0
UBR-PLAIN	150	101.0
ABR-EFCI	150	70.8

Results for high-speed ATM connections with 150-Mbps PCRs are shown in the third and fourth rows of Table 7.5. It is interesting to observe that while ABR-EFCI does improve TCP performance over the plain UBR service for low-speed connections, TCP performance actually deteriorates for high-speed connections.

Results in Table 7.5 clearly demonstrate that binary feedback is not effective for high-speed networking. Saito et al. [7] conducted simulation experiments with ABR-ER to assess the effectiveness of ER feedback for high-speed ABR connections. Table 7.6 shows TCP throughput over UBR-PLAIN and ABR-ER services for a 150-Mbps PCR (high-speed ATM connection) for 25 multiplexed VCs, indicating that ABR-ER achieves higher TCP throughput than UBR-PLAIN even for a high PCR.

From Tables 7.5 and 7.6, we can conclude that ABR-ER is superior to ABR-EFCI for high-speed networking. Because ABR-ER provides feedback on the degree of congestion, it is expected to achieve higher TCP performance. Table 7.7 [7] shows the performance of ABR-ER and ABR-EFCI for varying buffer sizes. A glance through Table 7.7 reveals that for limited buffer, ABR-ER achieves higher TCP throughput than ABR-EFCI; to achieve high throughput with ABR-EFCI, very large buffers are necessary.

Table 7.6
Performance for TCP Over ABR-ER for a 150-Mbps PCR
(25 VCs, 4,096 cell buffers)

Service	Aggregate TCP Throughput (Mbps)
UBR-PLAIN	101
ABR-ER	125

Table 7.7
Performance of ABR-ER and ABR-EFCI for Varying Buffer Sizes

Buffer Size (cells)	TCP Throughput (Mbps)	
	ABR-ER	ABR-EFCI
1,000	125	30
2,000	125	120
4,000	125	124

Based on ATM switches' buffering requirement for the complete prevention of cell loss, ABR-ER provides a more scalable solution than UBR-PLAIN. A simulation study by Kalyanaraman et al. [8] confirmed that ABR-ER switches with buffers equal to a small multiple of network diameter can guarantee no loss even for a very large number of VCs carrying TCP/IP. In contrast, UBR-PLAIN switches require buffers proportional to the sum of the TCP receiver window sizes.

7.4.4 ABR versus UBR enhancements

As ABR is a rather expensive solution to reduce cell loss and improve TCP performance, it is interesting to compare ABR performance with less expensive UBR enhancements, namely UBR-EPD and its variations. Fang and Lin [9] compared TCP performance over ABR-EFCI with UBR-EPD for varying buffer size and LAN and WAN distances and concluded that ABR-EFCI does not provide any significant performance gain over UBR-EPD. (Their performances are comparable.) In fact, for incorrectly tuned ABR parameters, ABR-EFCI's performance is worse than UBR-EPD.

There appear to be no explicit studies carried out to compare TCP performance over ABR-ER and UBR-EPD. However, since it has been established [7] that ABR-ER's performance is superior to that of ABR-EFCI (see Table 7.7), it is in order to conclude that ABR-ER will outperform UBR-EPD.

It is important to note that for sufficient buffers, there is no significant performance gain for one measure against another. The performance difference is more significant when buffers are limited. We sort various measures against the effect of ATM cell loss on TCP performance on the performance scale in Figure 7.13 for limited buffers in the ATM switches. We can see that the best TCP performance can be achieved with ABR-ER, while UBR-PLAIN achieves the worst. It should be emphasized though that the sorting in Figure 7.13 is based on limited buffers when cells are lost. For very large buffers (with no cell loss), TCP can achieve maximum possible throughput over UBR-PLAIN.

Figure 7.14 compares various UBR and ABR mechanisms on a cost scale. While ABR-ER provides the best TCP performance, it is the most expensive service. UBR-PLAIN is the least expensive service, and despite its poor performance, it may work as a motivation for many users—when it is available.

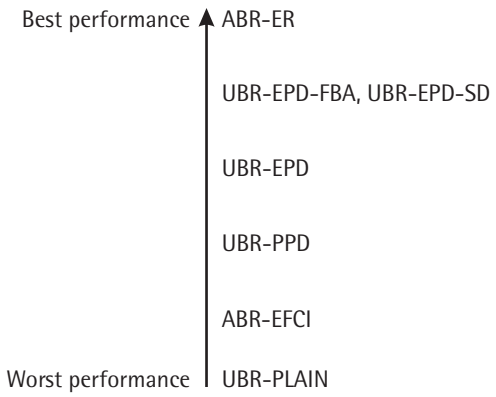


Figure 7.13 Comparing various UBR and ABR mechanisms on TCP performance scale.

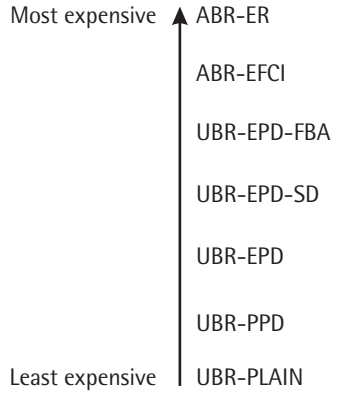


Figure 7.14 Comparing various UBR and ABR mechanisms on cost scale.

7.5 Forward error correction (FEC)

Errors in transmission links result in bits being received in error, and congestion in the network results in cells being dropped in the network. Automatic repeat request (ARQ) and FEC are two techniques used to recover from errored or lost cells. In the former method, the errored cells are retransmitted from the source.

FEC uses error-correcting codes to fix errors and recover errored packets. FEC may be operating at different layers of the network protocol

stack depending on the amount of data being corrected. For example, the data link layer generally uses FEC to recover errored bits in data link layer frames, whereas higher layers could be using FEC to recover lost cells. The principles of FEC used in both the layers are similar in concept. FEC is based on incorporating redundant bits (or cells) in the transmitted bits (or cells) to recover from errored bits at the data link layer or lost cells at the higher layer at the destination. FEC was developed by the telecommunications community and has not been very widely used in the Internet protocols.

7.5.1 Suitability of ARQ

ARQ involves retransmitting packets from the source. Cells lost for non-real-time unicast applications can be recovered using ARQ techniques. However, the cells lost in multicast communication have been found to be scattered and uncorrelated. ARQ is therefore very inefficient for multicast communication.

Retransmission incurs delay in receiving the retransmitted packets at the destination. ARQ techniques have therefore been found unsuitable for real-time applications having strict time constraints. As a result, FEC is almost exclusively used for error recovery in real-time applications.

7.5.2 Advantages/disadvantages of FEC

FEC has advantages over ARQ in the recovery of errored or lost cells in the sense that cells that are lost due to congestion in the network do not need to be retransmitted from the source. Instead, they can be recovered at the destination, thereby reducing the delay for real-time communication with time constraints.

FEC, on the other hand, suffers a number of disadvantages over ARQ, described as follows.

- It increases overhead in the packet due to the additional redundancy bits in the packet. A significant fraction of the transmission bandwidth is used to transmit the redundant bits used for implementing FEC.
- It increases the computational load on the source. The source needs to have sufficient computing power to execute the algorithm used to detect and correct errors.

- It is more effective in the case of random losses but is not very efficient in the case of losses that occur in bursts. This is because of the limited number of losses that can be tolerated per block of data.

The pattern of cell loss determines the complexity involved in recovering lost cells. For example, two cells lost in two different blocks are much easier to recover than two cells lost in the same block. Algorithms to disperse missing cells among many blocks are presented in [10].

7.5.3 Types of FEC

FEC operates by including redundant check bits to each block of data at the source [10]; lost bits can be reconstructed at the destination based on the check bits. The data bits and the check bits form codewords that are transmitted over a network. Although this section describes techniques for recovering bits, the same principles apply when recovering cells. In the case of cell-based FEC (used at layers higher than the data link layer), redundant cells are added with data cells to form the code. The major categories of FEC are described as follows.

- *Block codes:* A block code consists of k-bit blocks of data along with (n-k) check bits to form an (n, k) block code as shown in Figure 7.15. The minimum number of bit positions by which two codewords differ, called the Hamming distance, determines the error-correcting capability of the code. In general, for a code of distance d, the number of errors that can be corrected is given by $(d-1)/2$. Let's say that four valid codewords are given by:

  000000;
  000111;
  111000;
  111111.

Figure 7.15 An n-bit codeword with k data bits and $n - k$ check bits.

The Hamming distance between the codes is three, which means that single bit errors can be corrected.

- *Cyclic codes:* The cyclic codes are based on a principle similar to that of the block codes but use simple lateral shift to form the valid codewords. For example, if 0011 is a valid codeword, then the other codewords are obtained by shifting the bits of the above codeword to the left as shown below:

  0110;
  1100;
  1001.

Cyclic codes allow larger blocks of errors to be corrected than the noncyclic block codes.

- *Reed Solomon codes:* The Reed Solomon code is a type of cyclic code that is particularly useful for burst error correction (i.e., channels that have memory) of multiple errors. They are also useful for channels with a large number of input symbols.

7.5.4 Erasure codes

Bit-based error detection and correction schemes as described in Section 7.5.3 are used to detect and correct bit-level errors at the data link layer. For example, CRC is widely used to detect errors at the data link layer. After errors are corrected at the data link layers, the upper layers have mainly to deal with the missing cells, called *erasures*. Erasures result from uncorrectable errors at the data link layer or more frequently from packets lost in the network due to congestion.

Error-correcting codes at the upper layers are based on principles similar to those upon which the error-correcting codes at the data link layer are based (i.e., they are based on introducing redundancy in the transmitted data). The principle of erasure codes is to encode k cells into n cells of data (to be transmitted) in such a way so that the k cells of original data can be recovered at the destination from any subset of k cells out of the n cells of transmitted data [11, 12]. This implies that a minimum of k cells have to be received to be able to recover the original data as shown in Figure 7.16.

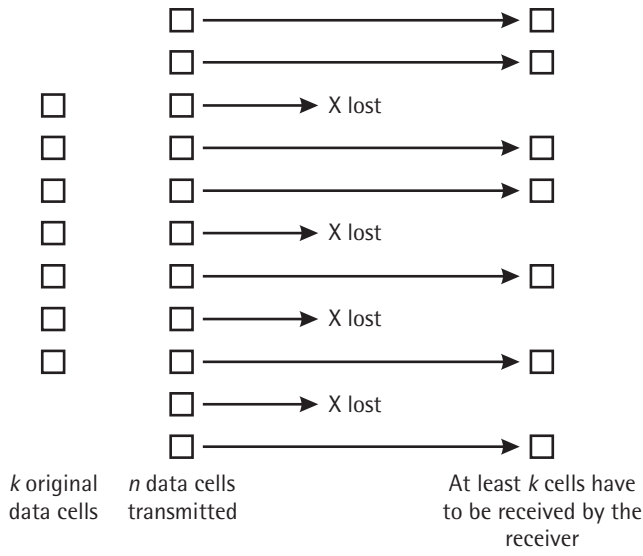


Figure 7.16 Illustration of erasure codes.

7.5.5 FEC in ATM networks

ATM cells consist of a five-byte header and 48 bytes of payload. To prevent misrouting of cells, the header is checked for errors at the intermediate switches. Out of the five bytes of header, one byte (called the header error correction) is used for correction of single-bit errors in the five-byte header field. Any cell whose header is found to be in error is discarded.

AAL1–4 use a sequence number for each cell that can be used to detect missing cells [13]. The first four bits of the cell payload form the sequence number in AAL1 and AAL2. In AAL3 and AAL4, the sequence numbers are in the third to sixth bits of the cell payload. Since AAL5 does not use sequence numbers, it is very difficult to detect lost cells when using AAL5.

The techniques used to recover lost cells using FEC at a higher level of the network are similar to those used to correct errors at the bit level as described in Sections 7.5.3. Figure 7.17 shows the coding matrix used by Ohta [14] to recover lost cells. A block consists of data cells organized into rows and columns. The last column is used for cell loss detection (CLD), and the last row consists of parity cells for data cell recovery. The rightmost cell of the last row serves as a parity cell for the cells in the last column (CLD cells).

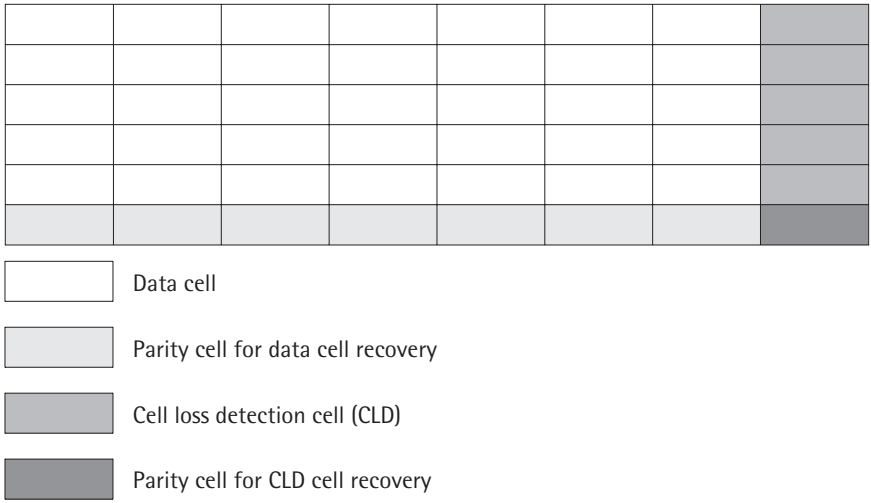


Figure 7.17 Coding matrix for recovery of ATM cells.

Kousa et al. [13] analyzed the performance of a *hybrid* scheme consisting of ARQ and FEC. In this hybrid scheme, FEC is performed on cells (using the coding scheme presented in [14]) followed by ARQ to recover the lost cells that cannot be recovered by FEC. This increases the efficiency of lost cell recovery over that which can be achieved by FEC only. They have studied two cases: The first case checks the header and the payload, whereas the second case checks only the header. They have concluded that when the BER of the channel is low (BER $< 10^{-7}$), correcting the payload was not advantageous when ARQ was used. This is because the small amount of error involved in the payload could be recovered by a small number of retransmissions, thereby avoiding costly overhead due to redundant cells required to correct the payload. However, for channels having high BER, checking the payload pays off the cost of redundancy very effectively. It was also observed that when both the header and payload are checked, the cell loss probability increases at a faster rate with increasing numbers of rows than in the case in which only the header was protected. Their second conclusion was that 17 columns in the coding matrix (see Figure 7.17) was the most appropriate, while the optimum value of the number of rows in the coding matrix depends on the channel BER, the number of hops between source and destination (note that link level error

correction is assumed at each hop), and the delay per hop. Similar results were obtained in [14] where it was found that the effective cell loss rate can be greatly reduced by having a coding matrix with 16 columns and 20 rows even when there are consecutive cell losses in the network. (Note that recovering from consecutive cell losses is much more difficult than random losses when FEC is used.)

7.5.6 Effectiveness of FEC

The effectiveness of FEC depends on the traffic burstiness and packet loss dependency. Assuming a certain channel error rate and a fixed level of redundancy in the code, the higher the burstiness of the traffic, the higher is the probability that information recovery will fail due to too many errored packets within a code block. This corresponds to an increased effective channel error rate at the bit level. FEC can significantly improve the effective CLR of a network. For example, it has been shown in [10] that FEC can result in a loss ratio (defined as the cell loss probability using FEC to the cell loss probability without FEC) of 10^{-3}, which represents a significant improvement in the recovery of lost cells. Although there has been considerable study of the performance of FEC at the cell level in ATM networks, the performance of TCP running over ATM using FEC is still an open issue that needs further study.

7.6 Summary

ATM is a packet-switched network based on statistical multiplexing. This may result in congestion and cell loss in the network during periods of overload. This chapter discusses four categories of solutions to avoid cell loss and minimize the impact of cell loss on TCP performance: (1) buffering at ATM switches, which works as a first line of defense against cell loss (the only defense for plain UBR service), (2) enhancements of UBR service using packet-based discarding schemes that are suitable for TCP/IP applications, (3) feedback-based congestion control used in the ABR service, and (4) FEC techniques to reconstruct lost cells at the destination.

If buffering is used as the sole measure against cell loss, as used in the plain UBR service, very large buffers (proportional to the number of TCP connections) are necessary to achieve good TCP performance. Packet-based UBR services, such as EPD, can significantly improve TCP

throughput over UBR for small buffers. Although effective for increasing throughput, UBR-EPD cannot achieve fairness; UBR-FBA and UBR-SD can improve fairness and achieve high throughput at the same time. ABR is a more complex and expensive solution to the cell loss problem and does not offer significant benefits over the simpler and less expensive UBR-EPD or UBR-FBA. FEC, meanwhile, can substantially reduce the effective cell loss rate and may be used in conjunction with UBR or ABR.

UBR-FBA is more complex to implement than UBR-EPD, and UBR-EPD is more complex to implement than UBR-PLAIN. It is not clear at this stage whether public ATM network service providers will support all flavors of UBR service, and if they do so, whether they will charge more for the more complex service. For private ATM networks, charging is not relevant.

FEC can be used at both the data link and upper layers. FEC is more effective in the presence of random losses than burst (consecutive) losses. The data link layer attempts to recover bit errors at the physical layer, while the upper layer is concerned with recovering from cell losses due to network congestion. FEC is based on transmitting redundant information along with data for the recovery of lost cells. The amount of redundancy determines the effectiveness of the algorithm and the amount of bandwidth wasted due to the redundant information.

References

[1] Oie, Y., et al, "Effect of Speedup in Nonblocking Packet Switch," *IEEE ICC '89*, Boston, MA, June 1989, pp. 410–414.

[2] Zhou, B., and M. Atiquzzaman, "A Performance Comparison of Buffering Schemes for Multistage Switches," *ICA3PP '95: IEEE First International Conference on Algorithms and Architectures for Parallel Processing, 19–21 April 1995, Brisbane, Australia, pp. 809–818.*

[3] Romanow, A., and S. Floyd, "Dynamics of TCP Traffic Over ATM Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 4, May 1995, pp. 633–641.

[4] Goyal, R., et al, "Improving the Performance of TCP Over the ATM-UBR Service," *Computer Communications,* Vol. 21, No. 10, July 1998, pp. 898–911.

[5] Fang, C., and A. Lin, "On TCP Performance of UBR With EPD and UBR-EPD With Fair Buffer Allocation Scheme," *ATM Forum Contribution*, 95-1645, December 1995.

[6] Stallings, W., *High-Speed Networks TCP/IP and ATM Design Principles*, Englewood Cliffs, NJ: Prentice Hall, 1998.

[7] Saito, H., et al, "Performance Issues in Public ABR Service," *IEEE Communication Magazine*, Vol. 34, No. 11, 1996, pp. 40–48.

[8] Kalyanaraman, S., et al, "Performance and Buffering Requirements of Internet Protocols Over ATM ABR and UBR Services," *IEEE Communication Magazine*, Vol. 36, No. 6, June 1998, pp. 152–157.

[9] Fang, C., and A. Lin, "TCP Performance in ATM Networks: ABR Parameter Tuning and ABR/UBR Comparisons," *Proc. IEEE Singapore International Conference on Networks*, Singapore, 1997, pp. 117–139.

[10] Shacham, N., and P. McKenney, "Packet Recovery in High-Speed Networks Using Coding and Buffer Management," *IEEE INFOCOM '90*, San Francisco, June 3–7, 1990, pp. 124–131.

[11] Rizzo, L., "On the Feasibility of Software FEC," *DEIT Technical Report* LR-970131, 1997.

[12] Biersack, E., "Performance Evaluation of Forward Error Correction in an ATM Environment," *IEEE Journal on Selected Areas in Communications*, Vol. 11, No. 4, May 1993, pp. 631–640.

[13] Kousa, M., A. Elhakeem, and H. Yang, "Performance of ATM Networks Under Hybrid ARQ/FEC Error Control Scheme," *IEEE/ACM Transactions on Networking*, Vol. 7, No. 6, December 1999, pp. 917–925.

[14] Ohta, H., and T. Kitami, "A Cell Loss Recovery Method Using FEC in ATM Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 9, Dec. 1991, pp. 1471–1483.

8

TCP/IP Over Switched Virtual Circuits

8.1 Introduction

IP is a connectionless protocol, whereas ATM is a connection-oriented network. An ATM connection has to be set up before any data communication can take place over an ATM network. Typically, an IP-ATM gateway connects a TCP network to an ATM network and manages the ATM connections (see Figure 8.1). As mentioned in Chapter 3, three types of connections can be set up in an ATM network: a PVC, a semipermanent VC (SPVC), and an SVC. The PVC and SPVC, which are set up manually, are usually held for a longer time than SVCs. Their lives range from days to months. They are usually suitable for emulating traditional leased lines used for a high volume of data transfers. With the PVC, a connection is already established before data is transferred; therefore, it has no connection setup overhead.

For interactive and bursty data communications, where the ATM connection is needed only temporarily, a PVC with QoS guarantees will cause unnecessary resource allocation when there is no data to be transferred. For such situations, SVCs are more appropriate. SVCs, which use signaling to open and close connections, can last from a few seconds to hours. The

Figure 8.1 Two TCP/IP subnets are connected through an ATM network using IP-ATM gateways.

ITU-T recommendation Q.2931 [1] specifies the signaling protocol at the user-network interface.

Issues related to TCP/IP communication with SVCs include how to manage the setup and closedown of SVCs to minimize the operational costs associated with SVCs and how to evaluate the performance of signaling software in terms of VC setup time. This chapter presents two SVC management techniques and develops models to analyze the performance of these techniques. Some benchmark tests to evaluate the performance of signaling are also discussed.

8.2 Operational cost of SVCs

The operational cost of an SVC depends on the following factors:

- *Setup rate:* This is the frequency of SVC setups and closedowns. Setting up an SVC requires signaling from the source to the destination and requires significant computing power on the part of the switches involved in the SVC. Consequently, if the SVC passes through a commercial carrier network, the carrier imposes a setup charge every time an SVC is set up. This is similar to the connection setup fee that is charged by telecommunications operators for every voice call. This cost, however, does not apply to private ATM networks.
- *Connect time:* An open SVC ties up resources in the switches of an ATM network. Such resources may include lookup table space, buffers, and bandwidth. Therefore, there is a cost associated with the duration of time for which the SVC is open. This is analogous to the per-minute charge imposed by telecommunications operators for voice calls.
- *Buffering cost:* When an IP packet arrives at the IP-ATM gateway, the packet may need to be buffered if the SVC is not already established

or the outgoing link is busy. The buffers hold up resources at the gateway and delay the packets at the IP-ATM gateway. These are accounted for by imposing a cost related to the amount of buffer being used. Buffering may also indirectly contribute to the cost of an SVC by delaying packets at the gateway.

Because of the bursty nature of data, the operational cost of an SVC can be optimized by properly managing the connection. The objective is to minimize the cost of the connection by determining a suitable mix of the above cost components. Section 8.3 describes two techniques to manage an SVC and determines the cost function for the two techniques.

8.3 SVC management

There are a number of ways an SVC can be managed to reduce the cost of an SVC. Closing the connection when there is no data in the buffer at the IP-ATM gateway can reduce the operational cost of an SVC. However, this will result in connection opening delays for subsequent arriving packets. Moreover, there is a connection setup cost associated with opening a connection. To reduce the cost, an SVC could be managed by leaving it open (in anticipation for more data) for some time (after it has been idle for a certain period of time) before it is closed. However, keeping the connection open too long will result in unnecessary operating cost and low utilization of the network. A second technique entails closing an SVC as soon as it is idle and setting up a new SVC connection later when there is a minimum number of packets at the IP-ATM gateway. Sections 8.3.1 and 8.3.2 describe these two techniques for managing an SVC.

8.3.1 Delaying the closedown (timer-based)

The cost of a connection can be reduced by monitoring the usage of a connection. As soon as a connection becomes idle, an inactivity timer can be started to measure the amount of idle time. If the connection has been idle for a certain period of time, as measured by the inactivity timer, the SVC could be closed. This technique has been proposed in RFC 1755 [2] and analyzed in [3]. The amount of time to leave an idle connection open before closedown (i.e., the value of the inactivity timer required to reduce the cost of an SVC) is an open question that is left to implementers to

decide. The value depends on, among other parameters, the traffic pattern arriving at the IP-ATM gateway. Because of the absence of suitable models to determine the value of the inactivity timer, ATM network administrators usually set the inactivity timer to a large value, up to 30 seconds. The large value of the inactivity timer may result in a waste of network resources and excessive SVC cost due to long idle times. For systems that limit the number of SVCs that can be opened simultaneously, idle SVCs can block the opening of new SVCs. In addition, the timer-based solution suffers from the disadvantage of requiring the maintenance of a large number of timers, especially for large systems. Section 8.4 discusses a queuing model that can be used to determine the optimal value of the inactivity timer to reduce the cost of an SVC.

8.3.2 Delaying the opening of an SVC (threshold-based)

The timer-based solution of managing an SVC requires users to manage timers and determine appropriate values for the timers. An alternative technique to manage an SVC has been proposed in [4], where an SVC is closed as soon as it becomes idle (i.e., there is no cost associated with the idling of an SVC). However, to reduce the cost and delay associated with the opening of an SVC, frequent opening of an SVC is avoided by letting a certain number of packets accumulate at the IP-ATM gateway before the SVC is opened. When the queue occupancy at the buffer reaches a threshold of K packets, a connection establishment procedure for opening the SVC is initiated. In this method, the control parameter, K, needs to be optimized to reduce the cost of an SVC. In queuing literature, this policy to control the service of queued events is sometimes referred to as the control operating policy (COP).

Sections 8.4 and 8.5 present performance models to analyze and optimize the performance of the above two SVC management schemes. These models are based on infinite buffers (no packet loss due to waiting) and Poisson packet arrivals at the IP-ATM interface. The implications of finite buffers and fractal (self-similar) arrivals are discussed in Section 8.6.

8.4 Performance model for a timer-based SVC

In the case of an SVC, a connection needs to be opened before data can be transferred. This results in an initial setup delay, but the utilization of the

connection is high because the connection can be closed down when there is no data to be transferred. During the connection setup, arriving packets will need to be stored in the buffers at the IP-ATM gateway. The buffers need to be dimensioned to avoid any possible loss of packets. The connection opening time, the duration of the connection, and the buffers required at the gateway contribute to the cost function of an SVC. The remainder of this section describes a delayed vacation model that allows us to study the behavior of VC setup rate γ, the average queue length L at the IP-ATM interface, and the amount of time VC remains open as a function of the control parameter timeout T.

8.4.1 Assumptions

Figure 8.2 shows the queuing model of an IP-ATM gateway buffer. It consists of a buffer fed by IP traffic; the buffer is served by a server that sends data over an SVC to the ATM network.

A queuing model, called the delayed vacation model (DVM) [5], is used to analyze the performance of the SVC. When the buffer is empty, the server waits for a certain amount of time before the SVC is closed and the server goes to vacation. This is equivalent to the vacation of the server being delayed by a certain amount of time; hence the name DVM. The model is based on the following assumptions.

- The buffer is infinite in size and is served by a FIFO queuing discipline.
- The arrival process to the queue is Poisson with a mean arrival rate of λ.
- The service time has a general distribution with an average rate of $1/\mu$.
- The server serves all packets in the queue exhaustively until the queue becomes empty. After serving the last packet in the queue, the server waits T time units anticipating the arrival of more packets. If

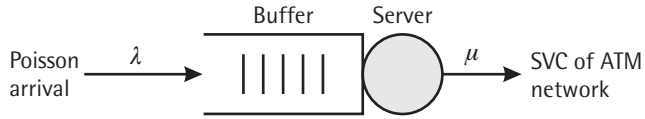


Figure 8.2 Modeling the buffer of an IP-ATM gateway.

no packet arrives within this T time units then the server is turned off (i.e., it goes to vacation). The concept is similar to going to a vacation after time T of the SVC becoming idle (hence the term delayed vacation). The DVM conserves bandwidth by closing the SVC when it is expected to be idle.

- If the server is not running (it is turned off) when a packet arrives, the server is turned on immediately. There is however, a service setup period of C time units. In other words, the server takes C time units from the moment it is turned on to the beginning of the service of the first packet waiting in the queue.

In the terminology of queuing theory, the DVM can be termed as an M/G/1 delayed vacation system with constant setup and delay times.

8.4.2 Server states

From the description in Section 8.4.1, it is apparent that the server will be in any one of the following four states at any instant of time:

- SERVE: The server is serving a packet.
- WAIT: Although the queue is empty, the server is ON, ready to serve and waiting for a packet to arrive.
- OFF: The server is turned OFF, and there is no packet waiting.
- SETUP: The server has been turned ON by a packet arrival but is not yet ready to serve; the server is doing some preservice work.

The above conditions translate to the state transition diagram shown in Figure 8.3. It is worth noticing that for $T = 0$, (i.e., if the server is turned OFF immediately after serving the last packet in the queue), the server goes to the OFF state directly from the SERVE state without entering the WAIT state at all. Therefore, for $T = 0$ the server has only three states, and the model reduces to a special case of an M/G/1 system under N-policy with the general setup time studied in [6].

8.4.3 Delayed vacation model (DVM)

Based on the known quantities (λ, μ, C, and T) described in Section 8.4.2, expressions for the following quantities are developed in this section to study the performance of a timer-based SVC.

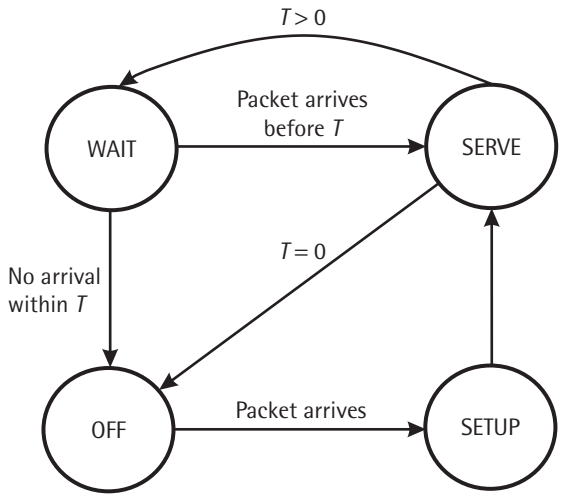


Figure 8.3 State transition diagram of delayed vacation model.

- Setup rate (λ), defined as the number of times the server enters the SETUP state per unit time;
- Average Delay (W) per packet, defined as the time spent in the queue plus the service time;
- Average number of packets (N) in the system;
- State probabilities—i.e., the fraction of time the server spends in the different states (WAIT, SERVE, OFF, and SETUP).

8.4.3.1 Setup rate (λ)

The system goes through two main cycles:

- The busy cycle serving customers (SERVE) or doing preservice work (SETUP);
- The idle cycle (OFF or WAIT) not doing any work.

The Idle cycle rate (β) is defined as the number of idle cycles per unit time as follows:

$$\beta = \frac{\text{Fraction of time spent in idle cycles}}{\text{Average length of an idle cycle}}$$

The numerator and denominator of the above equation for an M/G/1 system with general setup times are given [6] as:

$$\frac{1-\rho}{1+\lambda\overline{C}} \text{ and } \frac{1}{\lambda}$$

respectively, where $\overline{C}$ is the expected value of the setup time, and $\rho = \lambda/\mu$ is the utilization of the server. Since every busy cycle is preceded by an idle cycle, the rate of busy cycles is equal to the idle cycle rate β. The value of β depends on whether $T = 0$ or $T > 0$ as shown below.

- *Case 1:* $T = 0$: Every busy cycle starts with a setup of C time units. Hence the setup rate $\gamma = \beta$, and the expected setup time $\overline{C} = C$. Therefore, β for $T = 0$ is given by:

$$\beta|_{T=0} = \frac{\lambda(1-\rho)}{1+\lambda C}$$

- *Case 2:* $T > 0$: For the DVM ($T > 0$) some busy cycles start with a SETUP state (when the SVC was closed), while the rest do not. The SETUP states can, therefore, be classified into two types. One type of SETUP requires zero setup time, and the other type requires C time units of SETUP time:

  Scenario 1: SETUP states with zero setup times are preceded by idle cycles of length $I \leq T$ which has a probability of $1 - e^{-\lambda T}$ for exponential customer interarrival times.

  Scenario 2: The SETUP state with C units of setup time has a probability of $e^{-\lambda T}$. Therefore, the expected setup time for the DVM is $\overline{C} = Ce^{-\lambda T}$.

The idle cycle rate for the DVM ($T > 0$) is thus given by:

$$\beta|_{T>0} = \frac{\lambda(1-\rho)}{1+\lambda Ce^{-\lambda T}}$$

For $T > 0$, only idle cycles of duration $I > T$ will be followed by a SETUP state. Therefore, the setup rate λ is given by

$$\gamma = \Pr\left[I > T\right]\beta\big|_{T>0} = \frac{e^{-\lambda T}\lambda(1-\rho)}{1+\lambda C e^{-\lambda T}} \tag{8.1}$$

8.4.3.2 State probabilities

Let π_s, π_v, π_0, and π_w denote the probabilities of the server being in the SETUP, SERVE, OFF, and WAIT states respectively. Therefore, from (8.1) we get,

$$\pi_s = \frac{Ce^{-\lambda T}\lambda(1-\rho)}{1+\lambda C e^{-\lambda T}} \tag{8.2}$$

However, π_s and π_v can be obtained from [6] as

$$\pi_v + \pi_s = \rho + \frac{C\lambda e^{-\lambda T}\lambda(1-\rho)}{1+\lambda C e^{-\lambda T}} \tag{8.3}$$

Therefore, from (8.2) and (8.3) we get

$$\pi_v = \rho \tag{8.4}$$

Since the fraction of time spent in the SERVE state represents the utilization of the server, we observe that the utilization of the server for the DVM is $\rho = \lambda/\mu$ and is not a function of setup (C) or delay time (T).

The server goes through a busy cycle followed by an idle cycle which is again followed by a busy cycle and so on as mentioned earlier. Some of the idle cycles include an OFF state, while others do not. An idle cycle includes an OFF state if the duration of the idle cycle is greater than T. If an idle cycle includes an OFF state, then it must have spent time T in the WAIT state before going to the OFF state (the SVC was closed). If an idle cycle does not include an OFF state (SVC was idle but was not closed), then it spends an idle time $I \leq T$ in the WAIT state (before going to the SERVE state), where the length of I is exponentially distributed between 0 and T. The probability that an idle cycle includes an OFF state is therefore $e^{-\lambda T}$. Since π_w is the fraction of time spent in the WAIT state, π_w can be expressed as

$$\pi_w = e^{-\lambda T}\beta|_{T>0} + \beta|_{T>0}\, E(I)|_{I\le T} \tag{8.5}$$

where $E(I)|_{I\le T}$ is the expected value of I and can be obtained as

$$\int_0^T I\,\mathrm{pdf}(I)dI$$

Since I is exponentially distributed, we get

$$E(I)|_{I\le T} = \int_0^T I\lambda e^{-\lambda I}dI = \frac{1}{\lambda}\left[1 - e^{-\lambda T}(\lambda T + 1)\right] \tag{8.6}$$

Since all the state probabilities add to one, π_0 is obtained as

$$\pi_0 = 1 - \pi_s - \pi_v - \pi_w \tag{8.7}$$

8.4.3.3 Average delay, W

From the Pollaczek-Khinchin (P-K) formula [7], the average waiting time (W) of a packet in an M/G/1 system is obtained as:

$$W = \overline{X} + \frac{\lambda \overline{X}^2}{2(1-\rho)} \tag{8.8}$$

where $\overline{X}$ and $\overline{X}^2$ are the first and second moments of the service time, and ρ is the utilization of the server. Substituting

$$\rho = \frac{\lambda}{\mu}$$

the waiting time of a packet in the IP-ATM gateway buffer can be expressed as

$$W = \overline{X} + \frac{\lambda \overline{X}^2}{2\left(1 - \frac{\lambda}{\mu}\right)} \tag{8.9}$$

Applying Little's formula, the average number of customers, N, in the system is given by:

$$N = \lambda \overline{X} + \frac{\lambda^2 \overline{X}^2}{2\left(1 - \frac{\lambda}{\mu}\right)} \tag{8.10}$$

Packets arriving at the IP-ATM gateway when the buffer is empty can be classified into two categories:

- Normal packets that arrive at the system when the server is in WAIT, SERVE, or SETUP state;
- Exceptional packets that arrive when the server is in the OFF state.

The average service time for normal packets is

$$\overline{X}_n = \frac{1}{\mu}$$

(since the SVC is open when the packet arrives), and the average service time for the exceptional packets is

$$\overline{X}_e = \frac{1}{\mu} + C$$

(recall that the server requires C time units to open an SVC). If P_e is the probability that a packet is an exceptional packet, the first and second moments of the average time of the packets arriving at an IP-ATM gateway can be expressed as follows:

$$\overline{X} = P_e \overline{X}_e + (1 - P_e)\overline{X}_n = \frac{1}{\mu} + P_e C \tag{8.11}$$

$$\overline{X}^2 = P_e \overline{X}_e^2 + (1 - P_e)\overline{X}_n^2 \tag{8.12}$$

where $\overline{X}_e^2$ and $\overline{X}_n^2$ are the second moments of the service times of the exceptional and normal packets, respectively.

P_e is equivalent to the fraction of packets that require SVC setups. P_e is simply given by

$$\frac{\gamma}{\lambda}$$

Substituting

$$P_e = \frac{\gamma}{\lambda}$$

in (8.11) and (8.12) we obtain

$$\overline{X} = \frac{1}{\mu} + \frac{\gamma}{\lambda} C \tag{8.13}$$

$$\overline{X}^2 = \frac{\gamma}{\lambda} \overline{X}_e^2 + \left(1 - \frac{\gamma}{\lambda}\right) \overline{X}_n^2 \tag{8.14}$$

The second moments depend on the service time distribution. As an example, for deterministic service time

$$\overline{X}_e^2 = 2\left(\frac{1}{\mu} + C\right)^2, \overline{X}_n^2 = 2\left(\frac{1}{\mu}\right)^2$$

and the second moments can be obtained from (8.14) as

$$\overline{X}^2 = 2\left(\frac{1}{\mu^2} + \frac{\gamma C^2}{\lambda} + \frac{2\gamma C}{\mu\lambda}\right) \tag{8.15}$$

Given the service time distribution of packets, the second moments of the distributions can be calculated in a similar fashion.

A note on (8.9) and (8.10) for the derivation of *W* and *N* A further analysis [8] revealed that (8.9) and (8.10) are only valid for fixed-size IP packets, which yields to constant service time distribution for a given link

capacity. For variably sized IP packets, where the service time distribution is not constant, (8.9) and (8.10) provide only approximate results. The correct equation for W for any service time distribution is derived in [8].

8.4.3.4 Numerical results

The SVC setup rate (γ), SVC idling ratio (ξ), and average delay (W) as a function of the inactivity timeout period (T) are shown in Figures 8.4, 8.5, and 8.6, respectively. The SVC idling ratio is defined as the fraction of time an open SVC is idling on the average and is obtained from the ratio

$$\frac{\pi_w}{\pi_w + \pi_v}$$

The values used to generate the graphs are listed in Table 8.1.

The graphs show results obtained from the above model superimposed on simulation results. The close match between the analytical and simulation results verifies the accuracy of the above model. The simulation model

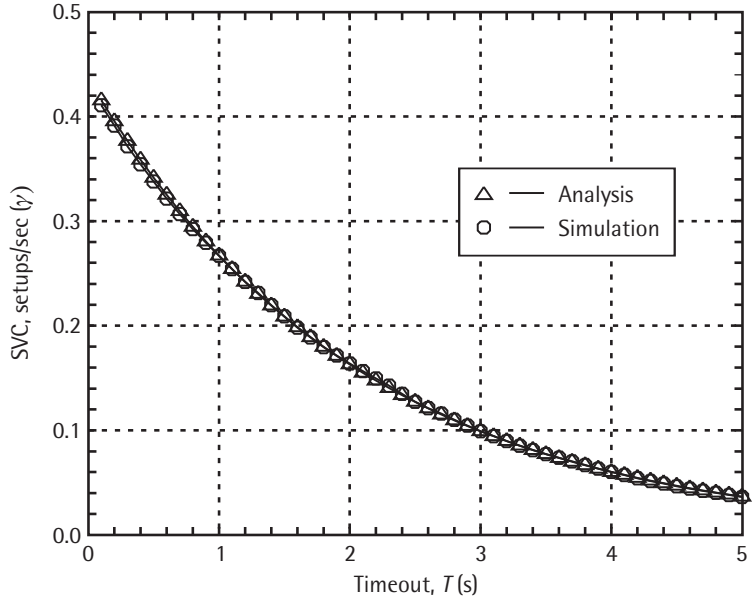


Figure 8.4 SVC setup rate versus inactivity timeout period.

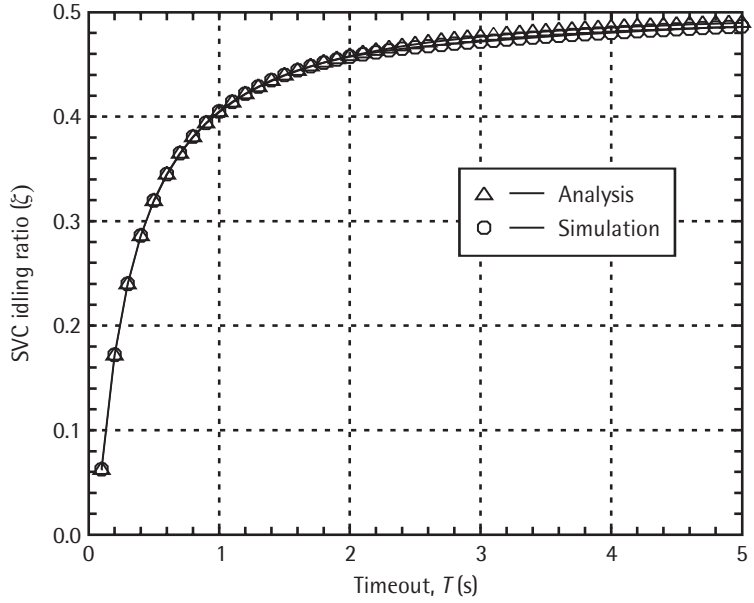


Figure 8.5 SVC idling ratio versus inactivity timeout period.

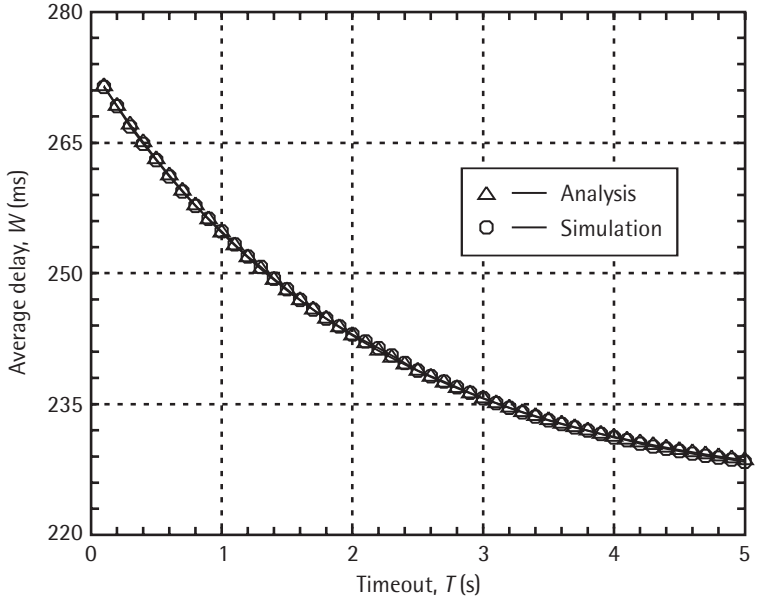


Figure 8.6 Average delay versus inactivity timeout period.

Table 8.1
Parameter Values for Figures 8.4, 8.5, and 8.6.

Parameter	Value
Mean arrival rate, λ	0.5 packets/s
Mean service rate, μ	4.72 packets/s
SVC setup time, C	50 ms

[5] consists of a Telnet session where a user is typing at the keyboard. The resulting packets are carried over the SVC. AAL5 has been used to carry the packets over the ATM network.

Figure 8.4 shows that as the inactivity timer increases, the SVC setup rate decreases exponentially. This is expected because the SVC is allowed to idle for a longer duration before being closed down, which results in fewer SVC setups. Consequently, as the value of the inactivity timer increases, the cost associated with SVC setup decreases at the expense of increased cost due to longer connection hold time.

An increased value of the inactivity timer results in longer idling times when there is no data to be sent. This is reflected in Figure 8.5 where the SVC idling ratio increases exponentially with an increase in the value of the inactivity timer.

A packet arriving at an IP-ATM gateway may encounter delay before it can be sent on the ATM network. If the SVC is open, the delay is due to queuing at the gateway. However, if the SVC is closed when a packet arrives, the SVC setup time contributes additional delay on top of the queuing delay. As the value of the inactivity timer is increased, the probability that an arriving packet finds the SVC as closed is reduced. This is reflected in Figure 8.6 where the average delay encountered by a packet decreases as the value of the inactivity timer is increased.

8.4.4 Optimizing the performance

The cost of an SVC based on an inactivity timer depends on the proper choice of the timeout value, T. The inactivity timer has to be appropriately set to minimize the cost of the SVC. The value of the inactivity timer depends on the traffic pattern arriving at the gateway. The cost of an SVC that is using the DVM can be expressed as [4]:

$$C_d(T) = A\gamma + BL + R\left(\rho + \frac{\pi_w}{\pi_w + \pi_v}\right), \qquad K = 1, 2\ldots \qquad (8.16)$$

where,

A = Cost of opening an SVC;

B = Buffering cost per unit time per IP datagram present in the system;

R = Cost per unit time for running the SVC;

L = Average queue length, defined as the number of IP datagrams in the queue at the IP-ATM interface including the one being transmitted over the SVC;

ρ = Load on the IP-ATM interface $\left(\rho = \frac{\lambda}{\mu}\right)$;

γ = SVC opening rate, representing the number of times an SVC needs to be opened per unit time.

Using the values of γ, π_w, and π_v obtained in Section 8.4.3, and the value of L from [8] we can obtain

$$C_d(T) = \frac{A\lambda e^{-\lambda T}(1-\rho)}{1+\lambda S e^{-\lambda T}} + B\left[\rho + \frac{\lambda^2 b_2}{2(1-\rho)} + \frac{\lambda S e^{-\lambda T}(2+\lambda S)}{2\left(1+\lambda S e^{-\lambda T}\right)}\right] + R\left[\rho + \frac{(1-\rho)\left(1-e^{-\lambda T}\right)}{1+\lambda S e^{-\lambda T}}\right] \qquad T \geq 0 \qquad (8.17)$$

where b_2 is the second moment of service time distribution. $C_d(T)$ for an M/M/1 queue can be obtained by replacing b_2 with

$$\frac{1}{\mu^2}$$

in (8.17).

The optimal value of T can be obtained by differentiating (8.17) with respect to T and setting it to zero as shown in [4]. It can be shown that the cost is minimized when the SVC is either closed as soon as the last packet from the IP-ATM gateway is transmitted ($T = 0$) or the SVC is always left open $T = \infty$ (corresponding to a PVC) depending on the SVC operating cost parameters A, B, and R. The implication of this result is that it may be difficult to find a fixed timeout value (fixed timeout refers to the situation where the same timeout value is used each time a timer is set) to minimize the SVC operational costs. Instead of fixed timers, adaptive timers (where a different timeout value is selected for successive timeouts) may be more effective for minimizing the SVC costs. Adaptive timers for managing SVCs are proposed in [3].

8.5 Performance model for threshold-based SVC

This section develops a model for the threshold-based SVC management mechanism, also referred to as COP. The model will enable one to determine the control parameter K of the COP.

8.5.1 VC setup rate gamma

For the COP, the SVC opening rate is equivalent to the number of times the queuing system becomes idle. The SVC opening rate can therefore be expressed as [6]

$$\gamma = \frac{\lambda(1-\rho)}{K+\lambda S} \tag{8.18}$$

8.5.2 Average queue length L

For an M/M/1 queue under COP with constant startup time (VC opening delay), the average queue length L is obtained as [9]

$$L = \frac{1}{2(K+\lambda S)}\left[K(K-1)+2K\lambda S+\lambda^2 S^2\right]+\frac{\rho^2}{1-\rho} \tag{8.19}$$

8.5.3 Optimizing performance

The cost function for the COP policy can be given by [4]

$$C_c(K) = A\gamma + BL + R\rho, \qquad K = 1, 2\ldots \tag{8.20}$$

where, A, B, R , and L are the same as defined earlier (in Section 8.4.4), and K is the queue threshold of the COP.

The first term in (8.20) represents the cost of opening an SVC; the second term accounts for the buffering cost; and the third term for the running cost of the SVC.

Substituting the expressions for γ and L from (8.18) and (8.19) into (8.20), and simplifying, we obtain the cost function for the COP as:

$$\begin{aligned} C_c(K) = {} & \frac{A\lambda(1-\rho)}{K+\lambda S} + \frac{B}{2(K+\lambda S)}\left[K(K-1)2 + K\lambda S + \lambda^2 S^2\right] \\ & + \rho\left(R + \frac{B\rho}{1-\rho}\right), \qquad K = 1, 2\ldots \end{aligned} \tag{8.21}$$

It is important to determine the optimum value of K for which the cost of operating an SVC under the COP is minimum. Let $\hat{K}$ be the optimal value of K. Therefore, $\hat{K}$ is obtained by differentiating $C_c(K)$ with respect to K and setting

$$\frac{dC_c(K)}{dK} = 0$$

We get

$$K_1 = -\lambda S - \sqrt{\lambda S + 2\frac{A}{B}\lambda(1-\rho)}$$

$$K_2 = -\lambda S + \sqrt{\lambda S + 2\frac{A}{B}\lambda(1-\rho)}$$

where K_1 and K_2 are the two roots of

$$\frac{dC_c(K)}{dK} = 0$$

K_1 is always negative and is not of our interest, as $C_c(K)$ is only valid for $K \geq 1$. For K_2, there are two cases:

- Case 1:

$$\frac{A}{B} \leq \frac{\lambda S(\lambda S - 1)}{2\lambda(1-\rho)}$$

  for which $K_2 \leq 1$, meaning $C_c(K)$ is an increasing function for $K \geq 1$; for this case $\hat{K} = 1$;

- Case 2:

$$\frac{A}{B} > \frac{\lambda S(\lambda S - 1)}{2\lambda(1-\rho)}$$

  for which $K_2 > 1$. It can be shown that $C_c(K)$ is a convex function for $K \geq 1$ with the minimum occurring at K_2. For this case, $\hat{K} = \lfloor K_2 \rfloor$ or $\lceil K_2 \rceil$.

Figures 8.7 and 8.8 show $C_c(K)$ for the above two conditions.

8.6 Infinite buffer and self-similar arrival

It should be noted that the performance models presented in Sections 8.4 and 8.5 are based on the assumptions that packets arrive according to a Poisson process and that there are enough buffers at the IP-ATM interface to hold all packets waiting to be transmitted over the ATM network. These two assumptions made it easier to develop the mathematical models, as basic queuing performance results for the Poisson process are readily available in the literature. In practice, however, it may not be accurate to assume that the traffic will be strictly Poisson. In fact, research shows [10] that IP traffic is self-similar and results in higher queuing delays than predicted by Poisson arrivals.

Queuing analysis for self-similar arrivals are extremely complex and are not readily available in the literature. Some preliminary simulation studies were performed in [11, 12] with network traffic trace collected

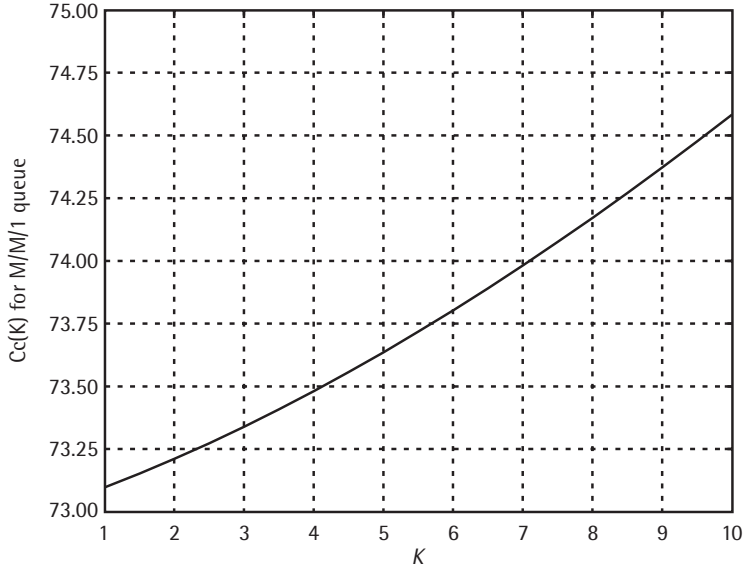


Figure 8.7 $C_c(K)$ for $K_2 > 1$ corresponding to $A = 20$, $\lambda = 100$, $\rho = 0.5$, $S = 0.5$, $B = 1$, $R = 55$.

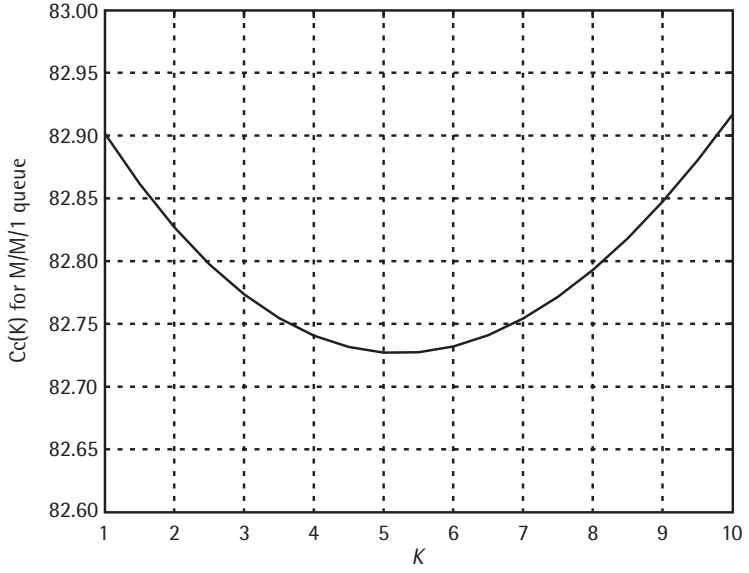


Figure 8.8 $C_c(K)$ for $K_2 \leq 1$ corresponding to $A = 30$, $\lambda = 100$, $\rho = 0.5$, $S = 0.5$, $B = 1$, $R = 55$.

from operational IP networks (these traces exhibit self-similarity) to assess the validity of the models presented in Sections 8.4 and 8.5 for real networks. Reference [11] presents results for the timer-based scheme, while [12] discusses results for the COP.

Simulation results in [11] show that the SVC operational cost as a function of timeout T either increases, decreases, or remains relatively constant depending on the choice of VC setup cost. This supports the result obtained from the Poisson-based model (see Section 8.4.4). Based on this limited study (only one network trace file was simulated), it appears that although self-similar traffic exhibits higher queuing delay than predicted by Poisson traffic, the Poisson-based SVC model may not be too inaccurate when predicting the trends of cost variables (e.g., SVC setup rate) as a function of the timeout value T. More comprehensive simulations are needed to verify the results presented in [11].

The performance study on queue threshold-based or the COP scheme in [12] was more comprehensive than [11]; it contains simulations of six trace files (instead of just one in [11]) all showing the self-similar property. The objective of the simulations was to investigate the behaviors of VC setup rate and queuing delay as a function of the control parameter K. (K is the queue threshold to initiate the VC setup process.) Simulation results demonstrate that in general, the Poisson-based COP model (as presented in Section 8.5) grossly underestimates the queuing delay and overestimates the VC setup rate. Therefore, the Poisson-based model cannot be used for quantitative analysis (for example, what would be the VC set up rate for $K = 3$?) of the COP scheme.

Simulation results in [12], however, revealed an important phenomenon; they show that in most cases, by slightly increasing the value of K, the VC setup rate can be decreased almost exponentially with only a linear increase in queuing length and queuing delay. This is in line with the results from the Poisson-based model presented in Section 8.5. Therefore, it appears that although the Poisson-based model is not good for quantitative analysis of the COP scheme, it can be used effectively to derive the optimum value of K that will minimize the operational costs of SVCs managed by the COP scheme.

For simplicity, the models presented in Sections 8.4 and 8.5 assumed enough capacity at the IP-ATM interface to hold all arriving packets while an SVC is being set up or packets are being transmitted over the SVC. While this may be true for some systems with very large buffers, in general

this assumption is not valid, as buffers are always finite in real systems, and there may be cases of buffer overflow during congestion. However, queuing analysis with finite capacity is very complex; the incorporation of parameters T (for timer-based SVC schemes) and K (for COP) make queuing analysis even more complex. In addition to the VC setup rate, queuing length, and queuing delay, a finite buffer model will have to derive a new performance parameter, the packet loss rate as a function of T or K.

No study has been reported to analyze COP under finite buffers. Niu and Takahashi [13] have developed a finite buffer model to analyze the performance of timer-based SVCs. This model allows one to derive the packet loss probability as a function of the buffer size and timeout parameter T, but no simulations with real traffic trace were reported to assess the effectiveness of this model.

8.7 Performance evaluation of signaling

Signaling is used by the software at the host to set up an SVC between hosts connected by an ATM network. The connection setup delay associated with the opening of an SVC contributes to the delay experienced by short-lived applications such as the World Wide Web, and distributed computing, modeling, and simulation. Excessive setup delay may contribute to cell losses at the IP-ATM gateway resulting in TCP retransmissions and degraded performance of the SVC. This has led to performance benchmarking studies of signaling in ATM networks [14].

There is currently no standard for benchmarking the performance of signaling to set up SVCs. Preliminary work in designing such a benchmark suite has been described by Niehaus, et al [14]. Tests have been carried out over LANs and WANs, spanning large ATM networks like MAGIC, ATDnet, and NRLnet. The benchmarking suite uses a call generator that is capable of setting up and releasing SVCs between specified hosts according to a specified behavioral profile.

SVC setup time is one of the important performance criteria of signaling. SVC setup time includes:

- The time taken by the switches to set up the call. It depends on the number of switches through which the call passes, the type of switches, the type of network (e.g., LAN and MAN) to which the switches are connected.

- The time taken by signaling software at the destination to make a decision to accept/reject the call. This time has been measured to be under 0.5 ms.

The factors that affect the performance of tests of SVCs include the following.

- *Load type*, which determines how the call requests are generated by the call generator and the duration of each call;
- *Call sequence:* The sequence of calls presented to the network before a new call requested is presented to the network;
- *Call rate:* The rate at which calls are generated;
- *Call duration:* The duration of calls can be adjusted to test the signaling software for how many active calls it can handle. For a given load, the longer the duration of calls, the more active calls are in the system.

8.7.1 Results

Test results for SVC setup times for LANs and WANs have been described in [14]. In the case of LANs, tests were carried out between a source to different destinations, which required traversing a varying number of switches as shown in Figure 8.9.

In the case of testing over WANs, a number of networks were connected such that the SVC between the source and destination had to cross several networks as shown in Figure 8.10.

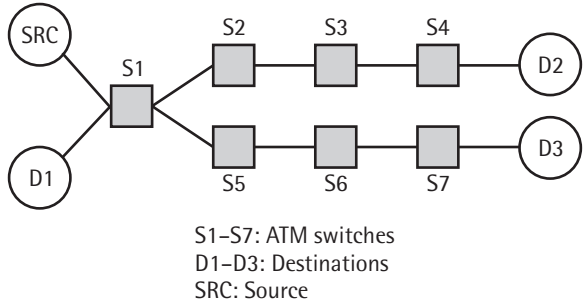


Figure 8.9 Topology for a source connected to a number of destinations in a LAN.

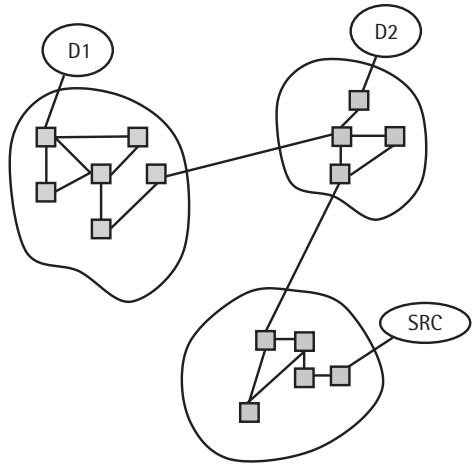


Figure 8.10 Topology for a source connected to a number of destinations over a WAN.

Table 8.2 shows the numerical results for the average call setup time in LANs and WANs for point-to-point and point-to-multipoint scenarios. The following observations are made from the performance test results:

- The call setup time is independent of the length of the test sequence.
- The setup time is linearly proportional to the number of switches spanned in the case of a LAN.
- The setup time for each hop is greater when the hops are between WANs.
- The setup times for point-to-point and point-to-multipoint are similar for destinations that are up to three switches away in the case of LANs. However, the setup times for point-to-multipoint for destinations more than three switches away are much greater than point-to-point connections.

8.8 Summary

With SVCs, ATM VCs are opened and closed dynamically using signaling. This chapter discusses two fundamentally different schemes for

Table 8.2
Average SVC Setup Time for Different Types of Connections Over LAN and WAN

Network Type	Type of Connection	Size of Network	Average Setup Time
LAN	P2P	SS	10–60 ms/hop
LAN	P2P	SPG	46–50 ms/hop
LAN	P2P	LPG	100 ms/hop
LAN	P2MP	SPG	50 ms/hop
LAN	P2MP	LPG	3,000 ms/hop
WAN	P2P	SS	20–800 ms/hop
WAN	P2P	LPG	75–100 ms/hop
WAN	P2MP	LPG	200–300 ms/hop

Note: P2P and P2MP represent point-to-point and point-to-multipoint connections, respectively. SS represents connections spanning a single switch (i.e., source and destination are connected to the same switch), SPG represents a small peer group of 10–20 switches, and LPG corresponds to large peer groups spanning a large number of switches (100–150).

managing SVC openings and closedowns between a pair of ATM hosts. The timer-based scheme delays the closedown while the queue threshold-based scheme delays the opening of an SVC. The chapter also presents mathematical models to analyze both schemes. These models help in determining the values of the control parameters to minimize the SVC operating costs. Finally, the chapter discusses the performance testing of SVC setup times for both LANs and WANs.

References

[1] ITU-T Recommendation Q.2931: B-ISDN User-Network Interface Layer 3 Specification for Basic Call/Bearer Control, March 1994.

[2] Perez, M., et al, "ATM Signaling Support for IP over ATM," *Internet RFC 1755*, IETF, February 1995.

[3] Keshav, S., C. Lund, S. Phillips, N. Reingold, and H. Saran, "An Empirical Evaluation of virtual circuit holding time Policies in IP-Over-ATM Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 8, October 1995, pp. 1371–1382.

[4] Hassan, M., and M. Atiquzzaman, "Virtual Circuit Management at IP-ATM Network Interface With Control Operating Policy," SPIE Conference on Performance & Control of Network Systems, (part of SPIE's Photonics East 1998, International Symposium on Voice, Video, & Data Communications), 2–4 November 1998, Boston, MA, pp. 434–441.

[5] Hassan, M., and M. Atiquzzaman, "A Delayed Vacation Model of an M/G/1 Queue With Setup Time and Its Applications to SVCC-Based ATM networks," *IEICE Transactions on Communications,* Vol. E80-B, No. 2, February 1997, pp. 317–323.

[6] Medhi, J., *Stochastic Models in Queuing Theory*, Boston: Academic Press, 1991.

[7] Bertsekas, D., and R. Gallager, *Data Networks*, Second Edition, Englewood Cliffs, NJ: Prentice Hall, 1992.

[8] Kitazume, H., and Y. Takahashi, "Waiting Time Distribution in a Single Server Queue With Vacation Setup/Closedown Times for IP Over ATM Networks," PMCCN '97 Workshop 3, Tsukuba, Japan, Nov. 21, 1997, pp. 58–65.

[9] Borthakur et al., "Poisson Input Queuing System With Startup Time Under Control Operating Policy," *Comp. Operations Research*, Vol. 14, 1987, pp. 33–40.

[10] Leland, W., M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Transactions on Networking*, Vol. 2, No. 1, February 1994, pp. 1–15.

[11] Hassan, M., R. Egudo, K. Power, and M. Atiquzzaman, "Effectiveness of Timer-Based Connection Management Mechanisms for IP/ATM Networks," Australasian Computer Science Conference (Computer Science '98), Springer-Verlag, February 1998, Heidelberg, Germany, pp. 489–497.

[12] Mifsud, T., and M. Hassan, "Performance Evaluation of Queue Threshold-Based Connection Management at IP-ATM Gateways," submitted to IEEE International Conference on Telecommunications, to be held in May 2000, Mexico. Available from: http://www.sd.monash.edu.au/~mahbub/PUBS/ict00trent.pdf.

[13] Niu, Z., and Y. Takahashi, "An Extended Queuing Model for SVC-Based IP Over ATM Networks and Its Analysis," IEEE GLOBECOM, 8–12 November, 1998, Sydney, Australia, pp. 490–495.

[14] Niehaus, D., A. Battou, et al, "Performance Benchmarking of Signaling in ATM Networks," *IEEE Communications Magazine*, August 1997, pp. 134–143.

9

End-to-End Traffic Management in IP/ATM Internetworks

9.1 Introduction

As discussed in Section 5.5, ABR control shifts any congestion from inside the ATM network to the IP-ATM edge device. The *congestion shift* phenomenon is depicted in Figure 9.1, where an IP LAN is connected to an ATM ABR service via an IP-ATM edge device. When congestion is detected inside the ATM network (somewhere along the ABR connection), the ABR feedback asks the IP-ATM edge device to reduce its transmission rate. This sudden reduction of transmission rate causes the buffer to fill up at the IP-ATM edge device (creates congestion in the IP-ATM edge device).

The traditional congestion-control algorithms used by conventional TCP/IP networks cannot effectively control the congestion in the IP-ATM edge device caused by sudden reduction of ABR bandwidth. Therefore, for IP/ATM internetworks, the benefit of ABR control cannot be fully realized unless there is some additional mechanism implemented between the IP-ATM edge device and the IP hosts to provide end-to-end traffic management.

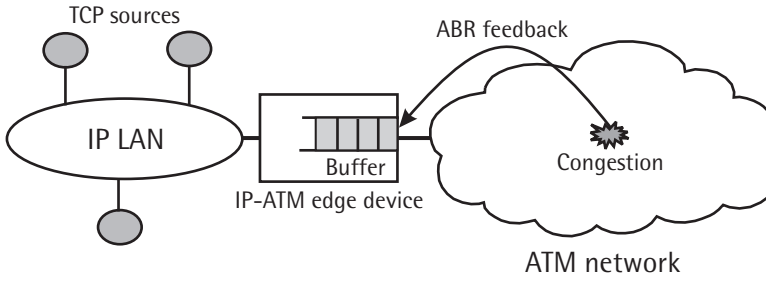


Figure 9.1 ABR control shifts congestion from inside the ATM network to the IP-ATM edge device.

Implementing end-to-end traffic management in IP/ATM internetworks has several benefits. First, the buffer overflow and packet loss at the gateway can be minimized, which in turn will increase the throughput for TCP traffic. Second, the buffering requirement at the gateway can be reduced as a result of better management of congestion. Smaller buffer means smaller queuing delay for packets traveling through the gateway, which, in turn, improves the performance for delay sensitive traffic, such as Telnet and Internet telephony.

A few end-to-end traffic management proposals have been or are being considered in the IETF for IP/ATM internetworks. One proposal differs from the other primarily in the way the congestion in the IP-ATM edge device is notified to the TCP/IP sources. This chapter describes several proposals to achieve end-to-end traffic management in IP/ATM internetworks by employing various congestion notification methods.

9.2 Adaptive packet discarding at IP-ATM edge device

TCP sources learn about the congestion in the network when packets are discarded (the retransmit timer for a discarded packet will eventually expire indicating congestion in the network). In existing IP gateways, the incoming packets are discarded when the buffer is full; this simple packet-discarding policy is usually referred to as the *drop-tail* policy. Such a drop-tail policy in an IP-ATM edge device is considered nonadaptive; the packet discarding never takes place until the buffer becomes full, ignoring the dynamics of the ABR.

When such drop-tail IP gateways are used as an edge device for IP/ATM internetworks, it takes a long time, until the buffer becomes full and packets are discarded, for a TCP source to learn about any sudden reduction of the ABR at the gateway. Due to this large feedback delay, TCP sources continue to transmit at a high rate even when the ABR at the gateway has been reduced drastically, causing severe congestion and packet loss at the gateway, which, in turn, reduces TCP throughput and increases packet delay.

Jagannath and Yin [1] proposed an adaptive packet-discarding algorithm for IP-ATM edge devices that aims to reduce the feedback delay of any ABR reduction at the gateway to the TCP sources by initiating packet discarding adaptively based on the ABR dynamics. With this algorithm, the gateway can discard a packet immediately when the ABR is drastically reduced, even when the buffer is not full. Since the gateway does not have to wait until the buffer is full, the adaptive packet-discarding algorithm effectively reduces the feedback delay of ABR reduction from the IP-ATM edge device to TCP sources. The feedback delay is further reduced by discarding the packets from the front of the buffer (the retransmit timer for the front packet will expire sooner, as it was started earlier, than that of the packet at the back).

Simulation results [1] show that for IP/ATM internetworks, the adaptive packet-discarding algorithm does achieve higher TCP throughput and lower delay than the simple drop-tail discarding policy in IP-ATM edge devices. However, there are several limitations for an end-to-end traffic management scheme relying entirely on the adaptive discarding algorithm in the IP-ATM edge device:

1. The scheme is not scalable in the sense that the feedback delay of the ABR reduction to TCP sources depends on the end-to-end distance; the longer the distance, the larger the TCP retransmit timer and the longer the feedback delay. The scheme aims to control longer term congestion in the IP-ATM edge device lasting on the scale of round-trip times; it is less effective in controlling shorter term congestion.
2. The scheme works only with TCP, as it relies on the window flow control algorithm implemented at the TCP layer. It will not be effective in controlling the rate of UDP, which does not implement any window flow control.

3. Since every packet discarded in the IP-ATM edge device has to be retransmitted by the TCP source, the TCP destination experiences large packet delays for the discarded packets. While large packet delays for some individual packets do not significantly affect the throughput of bulk data transfers (such as large file transfer), they significantly affect the performance of interactive connections, such as TELNET, for which the user is sensitive to individual packet delays. Therefore, the adaptive packet-discarding algorithm does not provide an effective end-to-end traffic management solution for interactive, delay-sensitive TCP applications.

It should be mentioned here that the Internet Draft [1] presenting Jagannath's adaptive packet-discarding algorithm expired in February 1998 without being advanced to the status of an RFC.

9.3 Explicit congestion notification (ECN) to TCP

The main drawback of the adaptive packet-discarding scheme [1] was the reliance on implicit congestion notification (ICN) to the TCP source by discarding packets at the IP-ATM edge device. Packet discarding increases packet delay, which in turn degrades the performance of interactive applications. Floyd [2, 3] explored the benefit of explicit congestion notification (ECN) for end-to-end TCP/IP networks using simulations. However, when used between an IP-ATM edge device and TCP sources in the IP subnet, the ECN method practically provides end-to-end traffic management for IP/ATM internetworks. Therefore, this chapter considers ECN as a viable solution for end-to-end traffic management in IP/ATM internetworks.

There are two ways the IP-ATM edge device can explicitly notify the TCP sources of any impending congestion without discarding packets in the gateway: (1) by sending ICMP source quench (SQ) messages back to the sources (backward notification) and (2) by setting an ECN bit in the IP header and having the destination return an ACK with an ECN bit in the header in response to an IP packet with an ECN bit (forward notification). Since the SQ method creates lots of extra traffic during congestion, only the second method implementing forward ECN (FECN) has been explored in [3].

9.3.1 Modifications to IP and TCP headers

ECN requires modifications to both IP and TCP headers for successful operation. The required modifications are described in detail in RFC 2481 [3]. These modifications are summarized in this Section.

Bits 6 and 7 in the 8-bit type of service (ToS) field of the IPv4 header were originally defined in RFC 791 [4] as unused; these two unused bits are utilized by ECN to convey congestion-related information at the IP layer. Bit 6, designated as an ECN-capable transport (ECT) bit, would be set by the data sender to indicate that this connection would like the gateways to indicate congestion by marking IP headers. The congestion experienced (CE) bit, which uses bit 7 of the ToS field, would be set by the gateways to indicate congestion to end hosts. Figure 9.2 shows the required modifications of the ToS field of the IPv4 header for implementing ECN.

ECN can also be implemented in IPv6. According to the latest definition of the IPv6 header in RFC 2460 [5], the traffic class byte in the header will correspond to the ToS byte in the current IPv4. The definitions of the IPv4 ToS byte and the IPv6 traffic class bytes are now superseded by the differentiated services (DS) field definition documented in RFC 2474 [6]. Since RFC 2474 defines bits 6 and 7 of the field as currently unused, ECN can be implemented in IPv6 and future IPv4 implementations without further modification.

The IP header carries congestion information only at the IP layer. For congestion notification at the TCP layer, ECN requires an ECN-echo flag in the TCP header so that the receiving TCP can inform the sending TCP that it has received a CE packet. The receiving TCP would continue to set ECN-echo flags in all ACKs, for both CE and nonCE packets, until it learns that the sending TCP has reacted to an ECN-echo packet and reduced its

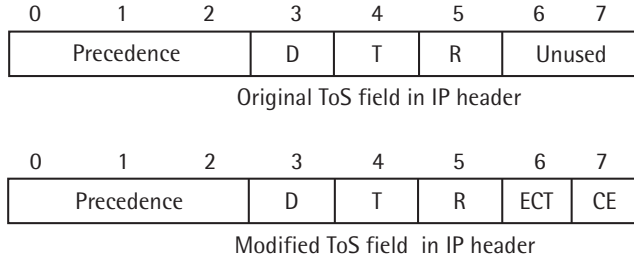


Figure 9.2 ECN modifications required in the ToS field of IPv4 (RFC 791) header.

congestion window. The congestion window reduced (CWR) flag in the TCP header is required for the sender to inform the receiver that the sender has reduced its congestion window. Bits 8 and 9 in the RESERVED field of the TCP header are used to designate CWR and ECN-echo flags, respectively [3]. Figure 9.3 shows the modified TCP header for ECN.

9.3.2 TCP's reaction to congestion notification

The ECN proposal [3] contains guidelines for TCP's response to ECN-echo ACKs (ACKs received with the ECN-echo flags set). According to these guidelines, the sending TCP treats an ECN-echo ACK the same as a retransmit time-out in a nonECN environment. That is, the TCP source reduces the congestion window to reduce the transmission rate.

At the same time, the guidelines are designed to make sure that the TCP does not respond to ECN-echo ACKs more frequently than necessary. More precisely, the ECN mechanism prevents the TCP source from reacting to congestion indication more than once every window of data (one window of data is cleared in one round-trip time).

9.3.3 Simulation of ECN

To evaluate the performance of the ECN scheme in TCP/IP networks, Floyd [2] simulated TCP traffic over a congested link (CBR link) interconnecting two IP gateways as shown in Figure 9.4. Traffic from both bulk-data transfer (e.g., large file transfer) and interactive applications (e.g., Telnet) were simulated. Figure 9.5 shows the average one-way TELNET packet delay (from the time the packet was transmitted at the source until it was received by the destination TELNET application) for

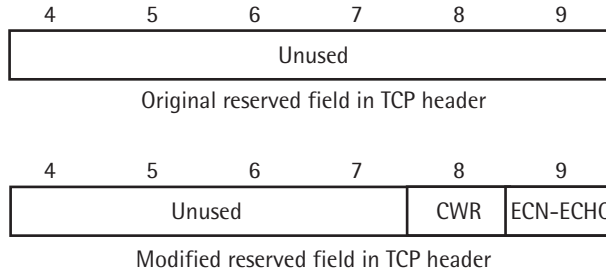


Figure 9.3 ECN modifications required in the RESERVED field of the TCP header.

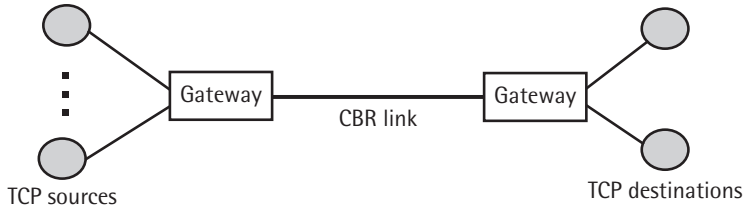


Figure 9.4 ECN simulation model.

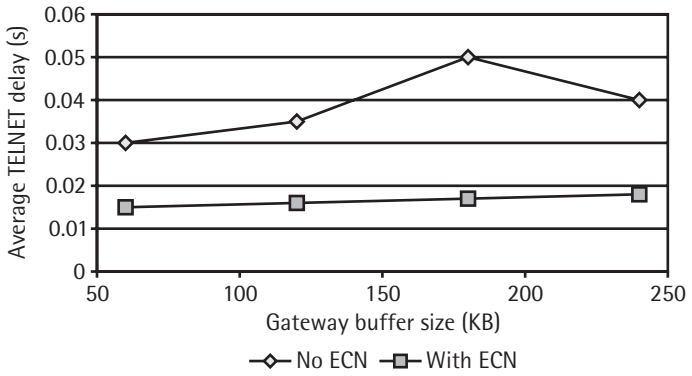


Figure 9.5 Average TELNET delay for ECN.

different gateway buffer sizes using the simulation parameters shown in Table 9.1.

Simulation results in Figure 9.5 confirm that the network delay for interactive traffic is reduced when ECN is implemented. Regarding throughput for bulk data transfers, Floyd's [2] simulation results suggest

Table 9.1
ECN Simulation Parameters

Simulation Parameter	Value
Total number of TELNET connections in each direction	10
Total number of bulk-data transfer connections in each direction	20
All link speeds	100 Mbps
Round-trip propagation delay between a source and destination	21–40 ms
Total simulation time	10 s

that ECN does not have a major impact on throughput except in short propagation delay and small buffer environment.

Later, another simulation study [7] of ECN was carried out to investigate the benefits of ECN. In this study, throughput was measured for FTP sessions with different file sizes. Table 9.2 shows the throughputs obtained for different file sizes for both ECN and nonECN connections. The results suggest that a moderate increase (about 20%) in TCP throughput can be achieved with ECN.

The simulations in [2, 7] were carried out with CBR leased line connecting two IP gateways. Such simulations cannot fully assess the effectiveness of ECN as an end-to-end traffic management solution for IP/ATM internetworks, where the IP gateways are connected by VBR ABR connections. Further investigation is necessary to evaluate the performance of ECN with ABR links.

9.3.4 Implementation of ECN

ECN has been implemented in a FreeBSD 2.2.2 platform at the University of California, Los Angeles (UCLA) [7]. The implementation is tested [7] in the UCLA laboratory with an experimental network configuration, as shown in Figure 9.6. There are two hosts, Mickey and Jerry, connected to two routers, Disney and Tom, respectively, via 100-Mbps links. The two routers themselves are connected to each other through a much slower

Table 9.2
TCP Throughput Increase for ECN

File Size (KB)	NonECN (Kbps)	ECN (Kbps)	% Gain
500	46.06	60	23.23
1,000	46.81	60.24	22.29
2,000	46.86	59.67	21.47

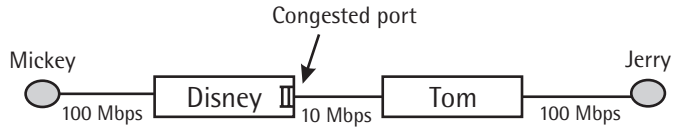


Figure 9.6 Network test-bed configuration for ECN.

link (10-Mbps). Mickey is used as a source and Jerry as a destination. The 10-Mbps interface in router Disney is used as a congested interface.

One TCP flow and one UDP flow were established from Mickey to Jerry. The congestion level at Disney was varied by varying the UDP traffic rate. Identical tests were conducted for both a nonECN environment and for when ECN is implemented in the router and in the host. Table 9.3 shows TCP throughput results for both ECN and nonECN TCP for two different congestion levels. These results confirm that TCP with ECN can achieve moderately higher throughput as indicated by the simulation results [7]. The level of benefit that can be achieved with ECN depends on the congestion level in the router. The higher the congestion, the more the benefit.

Only the throughputs for bulk data transfers were measured in these tests; the delay statistics were not collected. Currently, a number of other implementations of ECN are in progress. Floyd maintains a Web page [8] with links to a few implementations of ECN, completed or in-progress, along with links to ECN-related publications.

9.3.5 Limitations of ECN

ECN is superior to the adaptive packet-discarding algorithm in the sense that it reduces packet loss at the IP-ATM edge device, as it does not entirely depend on packet loss for congestion notification to TCP. Avoidance of every single packet loss has a direct impact on packet delay . However, like the adaptive packet-discarding scheme, the ECN scheme has the following limitations when used for end-to-end traffic management of IP/ATM networks:

1. ECN requires changes to both TCP and IP and hence limits its immediate deployment in the existing network infrastructure.

Table 9.3
TCP Throughput With ECN for Different Congestion Levels

UDP Rate (Mbps)	ECN Throughput (Kbps)	NonECN Throughput (Kbps)	% Gain
8.33	204.52	183.72	10.2
9.10	120.28	90.68	24.6

2. The scheme is not scalable in the sense that the feedback delay of the ABR rate reduction to TCP sources depends on the end-to-end distance; the longer the distance, the longer it takes for the ECN to provide the congestion feedback. The scheme aims to control longer term congestion in the IP-ATM edge device lasting in the scale of round-trip times; it is less effective in controlling shorter term congestion.

3. The scheme depends on TCP (or the transport layer), as it relies on the congestion-control algorithm of the TCP layer. It will not be effective in controlling the rate of UDP or other transport protocols that do not have ACKs and do not respond to network congestion. Every single transport protocol needs to be modified before it can benefit from ECN.

4. ECN relies on binary feedback; a single bit (CE bit) in the IP header is used by the gateways to notify the presence or absence of congestion to end hosts. With binary feedback, the *degree of congestion* cannot be notified accurately to the sources, resulting in longer oscillation and longer convergence time to steady state. The longer convergence problem with binary feedback schemes is a well-known one and has already been identified during the early experiments with the ABR control in ATM networks [9].

9.4 Backward and multilevel ECN

Salim et al [10] proposed two modifications to the ECN mechanism proposed in RFC 2481 [3]. The first modification is to provide backward ECN (BECN) using ICMP SQ messages, and the second one is to accommodate multiple levels of congestion feedback in the payload of ICMP messages. While the first proposed modification does improve the response time, it was already considered in the original ECN proposal by Floyd [2] but was not pursued further because SQ messages generate extra traffic during congestion.

The goal of the second proposed modification to ECN is to reduce the convergence time (to reach steady state) by providing more than two levels of congestion notification to sources. With multiple levels of notification,

the sources can take different levels of congestion control actions. No simulation or experimental results were reported for Salim's proposed modifications to ECN.

Section 9.5 describes an end-to-end traffic management scheme based on IP rate control, which aims to eliminate the drawbacks of the adaptive packet-discarding and ECN schemes.

9.5 Host gateway rate control protocol (HGRCP)

The primary limitations of the ECN mechanism proposed in RFC 2481 and currently being discussed with much interest in the Internet community are: (1) that it uses binary feedback, (2) that it relies on FCN, and (3) that it depends on TCP-like adaptive transport layer protocols. The first limitation leads to longer convergence to steady state; the second one makes it less effective for high delay-bandwidth links; and the third one reduces its applicability in a heterogeneous environment with both adaptive and nonadaptive transport layer protocols.

Hassan [11] proposed the host gateway rate control protocol (HGRCP), an end-to-end traffic management scheme for IP/ATM networks that addresses each of the above limitations of ECN by utilizing an ABR-style rate control mechanism at the IP layer. HGRCP is based on ER feedback, uses BECN, and implements the source rate control at the IP layer, eliminating the need to rely on the complicated congestion-control algorithms at the transport layer.

HGRCP is similar to the ER ABR flow control mechanism. The gateway implements an algorithm that periodically monitors the current ABR and the queue size to compute the ER for each contending source. The ER is then notified to the IP source hosts on the LAN using ICMP messages.

With HGRCP implemented in the IP LANs, there is ABR control in the ATM subnet and IP rate control in the IP subnet. While ABR control shifts the congestion from inside the ATM subnet to the IP-ATM edge device, IP rate control further pushes the congestion from the IP-ATM edge device right to the IP end hosts proving a true end-to-end solution to congestion management in the IP/ATM internetworking environment. For HGRCP-capable IP LANs, Figure 9.7 illustrates the relay of congestion from within the ATM network to the IP-ATM edge device and from the IP-ATM edge device to the hosts in the IP LANs.

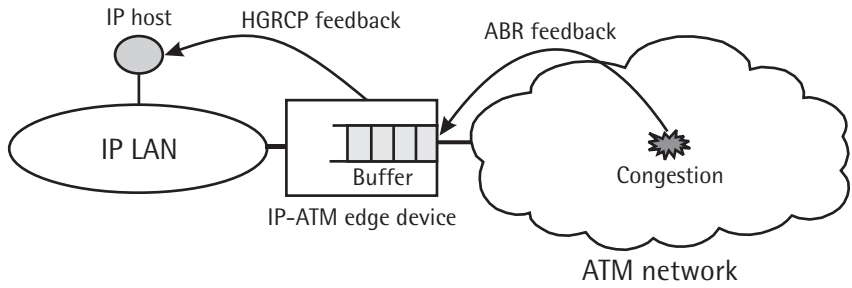


Figure 9.7 Congestion relay from inside an ATM network to the ATM edge and from the ATM edge to the end host.

9.5.1 Rate computation algorithm

The IP-ATM gateway periodically (at regular intervals of T seconds) calculates the effective aggregate input rate R to the gateway that will keep the current queue length q at a prespecified threshold Q. The gateway buffer is modeled as a linear feedback control system that yields,

$$R = ACR + K(Q - q) \tag{9.1}$$

where K is the control gain and ACR is the current ABR. The rate control system of (9.1) has the following properties [12]:

1. The control mechanism can be analyzed using classical control theory, and the control parameters can be selected to guarantee the stability of the control system. For example, using control-theoretic analysis it was shown [12] that a very stable system is achieved by selecting $K = 1/T$.

2. Because the current ABR bandwidth (ACR) is feed-forwarded, the control mechanism is not only *reactive*; it also becomes *proactive*. The feed forward of ACR makes the control system robust to ABR bandwidth variations and can guarantee a low queue length variance around the prespecified threshold Q. The low variance leads to smaller buffer requirement, smaller delay, and delay variance at the IP-ATM gateway.

3. It has a single control parameter (gain K) that reduces complexity in selecting optimal parameter values.

The gateway maintains a variable N for the number of LAN hosts competing for ABR bandwidth, which is obtained as the number of hosts that transmitted one or more packets through the ABR connection in the last W seconds. At the beginning of each T-second interval, the gateway obtains a single value for ER as

$$\mathrm{ER} = \max(0, R) / N$$

By default, HGRCP allocates this ER (equal allocation) to all competing hosts. However, the network administrator may choose to implement a more sophisticated rate-allocation algorithm to suit the need of certain servers and high-priority hosts. Section 9.5.2 explains how ER is notified to LAN hosts.

9.5.2 Rate notification

If the current ER value calculated is different than the previous ER value (ER value calculated T seconds earlier), the gateway notifies all the hosts on the LAN of this new ER using an ICMP message broadcast on the LAN. The broadcast is used to avoid loading the LAN with ICMP messages. Since almost all legacy LANs support hardware broadcasting, this provides an efficient way of implementing ER BECN. Given the T, one can easily calculate the ICMP load created on the LAN due to HGRCP. For reasonable values of T, it can be shown [11] that the ICMP overhead remains below 1% of the total LAN bandwidth for most practical LANs.

9.5.3 Rate control at the IP layer

In addition to the original queue, the IP host maintains several rate-managed queues, each for a different ABR queue (for gateways with multiple ABR links to multiple destinations, there is more than one ABR queue in the gateway) in the IP-ATM gateway, as shown in Figure 9.8. Rate-managed queues can be implemented by servicing the packets from the queue at a given rate.

If a rate-managed queue is full, the TCP process trying to add a packet to the queue will block, which will, in turn, block the application using the

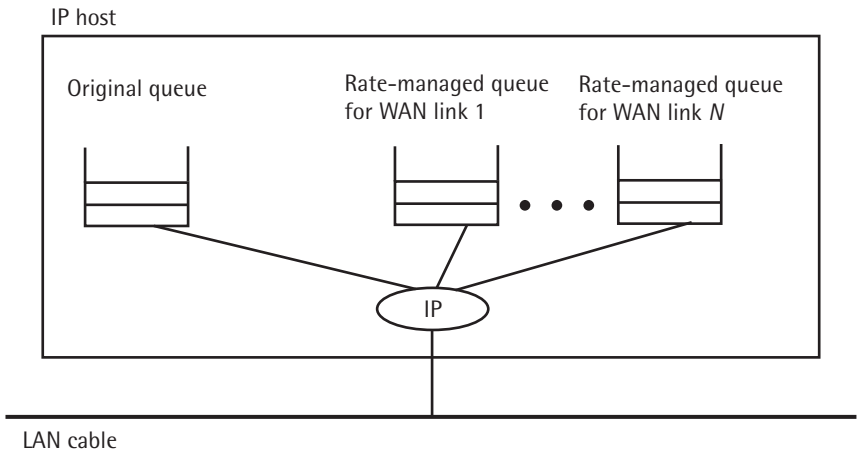


Figure 9.8 Rate control at IP host using rate managed queues.

TCP process. Therefore, the applications will not be able to pump data into the network during congestion.

9.5.4 Modification to IP header

Initially, a host does not know in which rate-managed queue it should put a packet for a given destination, and hence it puts the packet in the original queue. To indicate to the gateway whether the packet is coming from the original queue or from a rate-managed queue, the host makes use of an unused bit, designated as Q-bit, in the IP header. The currently unused bit 6 in the IPv4 ToS field is suggested for the Q-bit. The Q-bit is set by the host only if it is coming from rate-managed queues; otherwise it is left as zero. Figure 9.9 shows the HGRCP modification requirements for the ToS field in the IPv4 header. Like ECN, HGRCP can also be implemented in IPv6 and future implementations of IPv4 without further modification, as bit 6 of the DS byte, which replaces the traffic class byte in the IPv6 header and the ToS byte in the IPv4 header, is also defined as currently unused.

If the Q-bit is not set, the gateway notifies the host of the ABR queue identification for the destination network of the IP packet using a new ICMP message. Once the host receives this notification, all subsequent packets to this destination are put to a rate-managed queue corresponding to the ABR queue identification received in the ICMP message from the gateway.

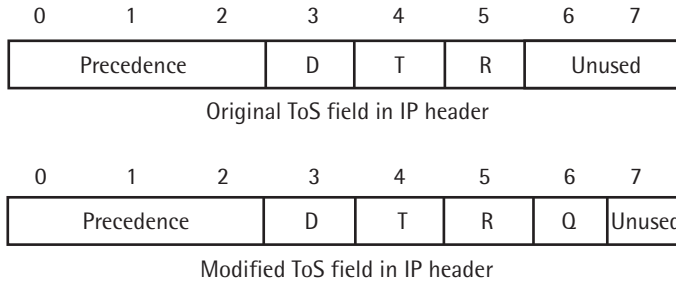


Figure 9.9 HGRCP modifications required in the ToS field of IPv4 (RFC 791) header.

The gateway clears the Q-bit when an IP packet leaves the gateway. This way, the Q-bit has significance only in the local LAN and can be reused for some other purposes in the global Internet.

9.5.5 New ICMP message types

HGRCP uses a new ICMP message, called a rate notification (RN) message, to notify the ERs to the sources. Many ICMP message types currently remain unused. HGRCP uses an unused message type, TYPE 20, to identify RN messages.

For IP-ATM edge devices with multiple ABR links to multiple IP LANs (a typical scenario for large corporations with many LANs in many cities), a control mechanism is needed for each queue associated with each ABR link. This makes the congestion notification format in the broadcast ICMP message slightly more complicated, as it now needs to identify a given queue in the IP-ATM edge device; there may be different ERs for different queues. Figure 9.10 shows the format of an ICMP RN message for gateways with multiple IP links. The Q_COUNT field tells how many (Qid, ER) pairs follow this field.

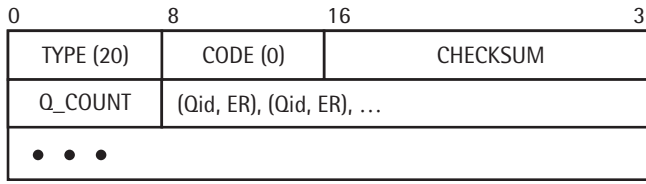


Figure 9.10 ICMP RN message format for gateways with multiple ABR links.

HGRCP uses TYPE 21 to identify the new ICMP message type, called a queue notification (QN) message, used by the gateway to notify a host of the ABR queue identification of a given IP destination. Figure 9.11 shows the format of an ICMP QN message.

9.5.6 Simulation of HGRCP

A detailed simulation study [11] was conducted with conventional TCP/IP and TCP/IP with HGRCP using a reference network configuration, as shown in Figure 9.12. To evaluate HGRCP against theworst case of ABR changes, the ABR bandwidth was controlled by an ON-OFF VBR source with exponentially distributed ON and OFF periods. The ABR remains at maximum (PCR) when the VBR source is ON and switches to minimum value (MCR) when the VBR goes to OFF state. Simulation configuration and parameters are summarized in Table 9.4.

Figure 9.13 shows the queue length variation at the IP-ATM gateway as a function of time. For conventional TCP/IP, queue length increases sharply at the sudden reduction of ABR bandwidth. In contrast, queue length is tightly controlled by HGRCP and kept close to the prespecified threshold value of 100 cells throughout the simulation.

The maximum queue length graph in Figure 9.14 illustrates another advantage of HGRCP. With conventional TCP/IP, the buffer size required to guarantee zero-loss increases almost linearly with the number of TCP connections competing for ABR bandwidth. However, with HGRCP, the buffer size required is not a function of the number of TCP connections.

The effectiveness of HGRCP in reducing the packet delay is demonstrated by the average queuing delay graph shown in Figure 9.15. The delay increases almost linearly with conventional TCP/IP as the number of TCP connections increases; however, the delay is tightly controlled by HGRCP.

Since conventional TCP/IP requires very large buffers to accommodate any dramatic decrease in ABR bandwidth, it will result in packet loss if sufficiently large buffers are not implemented in the IP-ATM gateway.

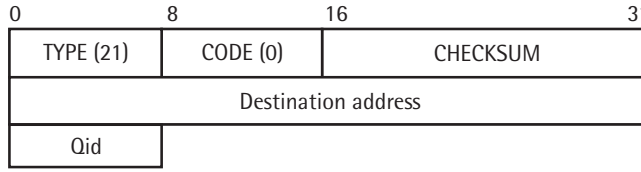


Figure 9.11 ICMP QN message format.

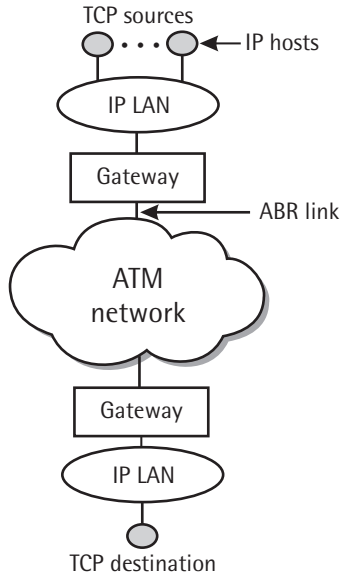


Figure 9.12 HGRCP simulation model.

Table 9.4
Simulation Parameters for HGRCP

Simulation Parameter	Value
LAN speed	10 Mbps
LAN frame format (for overhead purposes)	Ethernet
Number of TCP connections competing for ABR bandwidth	Variable (1–8)
ABR PCR	23,585 cells/s (10 Mbps)
ABR MCR	235 cells/s (1% of PCR)
ATM adaptation layer	AAL5
Round-trip delay in ATM network	60 ms
Mean ON period for background VBR source	100 ms
Mean OFF period for background VBR source	400 ms
HGRCP sampling interval T	10 ms
HGRCP controller gain K	100
HGRCP queue threshold Q	100 cells
HGRCP active host detection interval W	150 ms

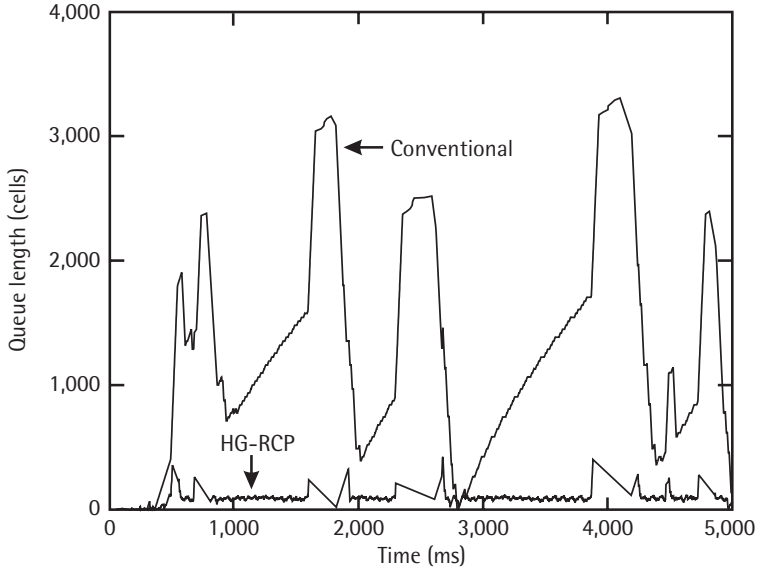


Figure 9.13 Queue length as a function of time for conventional TCP/IP and TCP/IP with HGRCP.

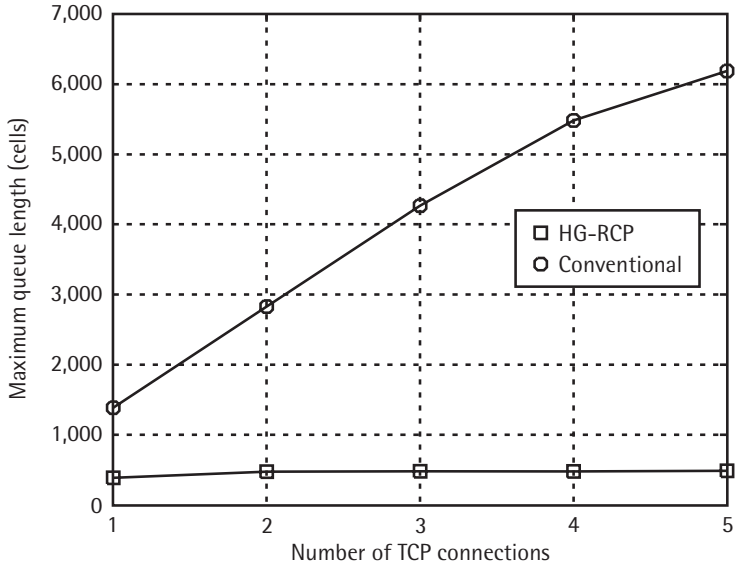


Figure 9.14 Maximum queue length for conventional TCP/IP and TCP/IP with HGRCP.

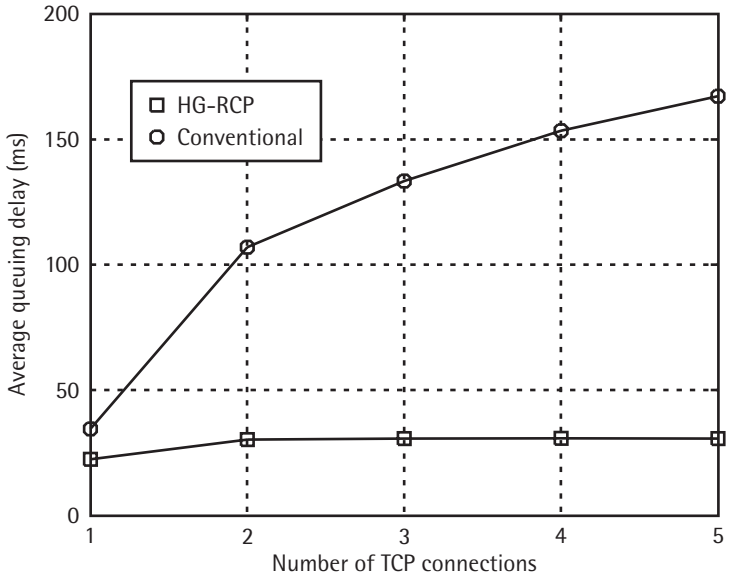


Figure 9.15 Queuing delay with conventional TCP/IP and TCP/IP with HGRCP.

Such packet loss will also affect TCP throughput. The impact of buffer size on TCP throughput is captured in Figure 9.16 (for eight competing TCP connections), which demonstrates that with conventional TCP/IP, TCP throughput can be severely reduced by dramatic reduction in ABR bandwidth unless large buffers are implemented in the gateway. However, HGRCP can effectively protect TCP throughput without requiring large buffers.

9.5.7 Implementation of HGRCP

HGRCP was implemented on a Linux kernel version 2.0.34, and some preliminary tests were conducted [13] at the network laboratory of Monash University. The testing procedure and the preliminary results obtained from the experimental tests are summarized in this section.

9.5.7.1 Test-bed configuration and test procedure

Figure 9.17 shows the network test-bed configuration for HGRCP. There are two 10-Mbps Ethernet segments, EtherA and EtherB, connected via two gateways with a serial link (a CBR link connected to serial

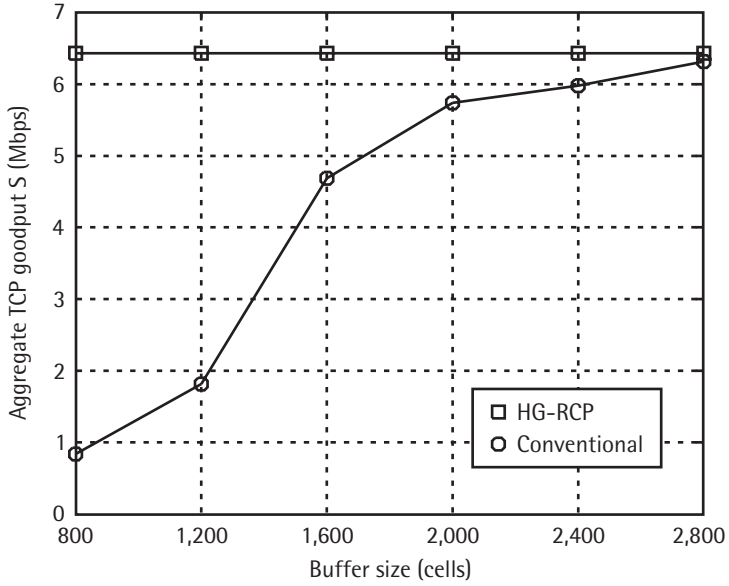


Figure 9.16 Impact of buffer size on TCP throughput for conventional TCP/IP and TCP/IP with HGRCP (for eight simultaneous TCP connections).

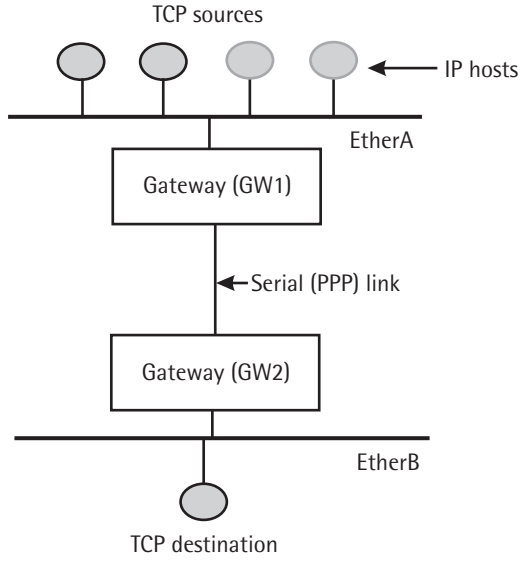


Figure 9.17 Network test-bed configuration for HGRCP.

communication ports of the gateways) between them using the PPP. The serial link speed was configured to the maximum value of 115,200 bps, which is roughly equivalent to a 128K ISDN link used by many corporate LANs to connect to the external WAN. Note that the above configuration tests the usefulness of HGRCP for LAN interconnection over slow, congested links; it does not test HGRCP for IP/ATM networks.

Four HGRCP-capable hosts on EtherA send large files to a single standard Linux host on EtherB using FTP. The gateway connected to EtherA (GW1) could be switched between HGRCP and standard implementations to repeat the same tests for both implementations. The values of HGRCP parameters used in the tests are shown in Table 9.5.

9.5.7.2 Test results

Figures 9.18 and 9.19 show the maximum and average queue lengths as a function of traffic load at GW1. With HGRCP, the maximum queue length remains stable at 20 packets, and the average queue length stabilizes at the prespecified threshold value of 10 packets ($Q = 10$ in Table 9.2). In contrast, the standard implementation of TCP/IP fails to keep the queue length at a stable value; the queue length increases almost linearly with an increase in the number of TCP connections as predicted by earlier simulations [11].

9.5.8 Limitations of HGRCP

Although HGRCP appears to provide an effective solution to the gateway congestion problem in IP/ATM networks, it has several limitations.

- *End-system dependence:* In addition to the gateway, all hosts in a given site need to be upgraded to deploy HGRCP. This prevents immediate deployment of HGRCP in the existing networking infrastructure. However, this limitation is not specific to HGRCP, any

Table 9.5
HGRCP Parameter Values Used in the Tests

HGRCP Parameter	Value
Sampling interval T	0.5 s
Controller gain K	2
Queue threshold Q	10 packets
Active host detection interval W	1 s

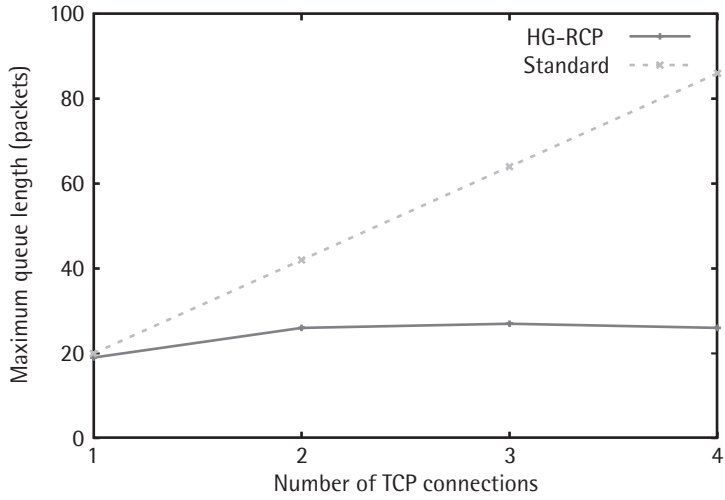


Figure 9.18 Maximum queue length as a function of the number of TCP connections.

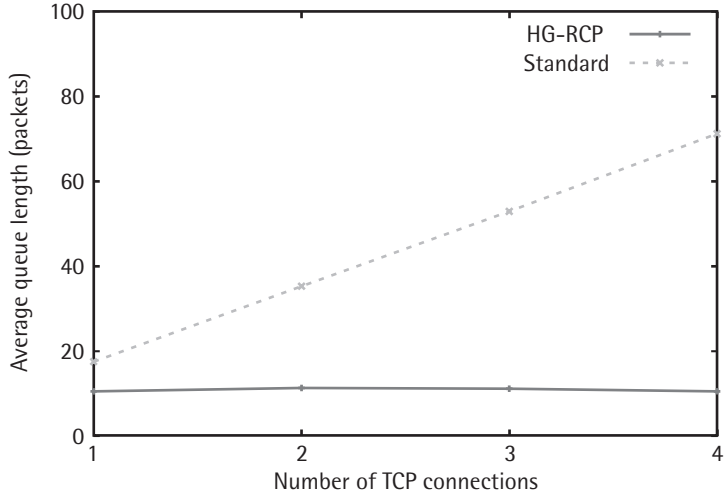


Figure 9.19 Average queue length as a function of the number of TCP connections.

mechanism, including ECN (RFC 2481), that requires changes to an IP or TCP header or protocol "suffers" from this limitation.

- *Aggregate rate control at layer 3:* Rate allocation and control is based on aggregate flows in a given IP host, not on individual (TCP or UDP) flows. If there are two TCP flows, one carrying FTP and the other Telnet data, originating in the same host and terminating at the same destination network (but possibly in different destination hosts), both flows will be treated as a single IP source and will be rate-controlled using a single FIFO queue at the IP layer in the originating host. Such aggregate rate control at the IP layer makes it difficult to allocate arbitrary rates to individual TCP connections.

9.6 TCP rate control

The second limitation of HGRCP (i.e., the lack of rate control on individual flows) can be addressed if the rate control is applied at layer 4 instead of layer 3. TCP is the layer 4 protocol in TCP/IP stack. By applying rate control at the TCP layer, each TCP connection can be allocated different rates.

Finding a solution to the first limitation of HGRCP (the end system dependence) is more challenging. The challenge comes from the fact that TCP is a layer 4 protocol and hence resides only at end systems. Introducing rate control at the end system process, without tampering with the existing process implementation, requires some clever solutions.

9.6.1 ACK-bucket control

ACK-bucket control [14, 15] is an innovative layer 4 rate control scheme to control the transmission rate of individual TCP connections without modifying the TCP header or the protocol. It is an end-system transparent solution that can be implemented without any changes to the existing TCP/IP implementations at the end hosts.

ACK-bucket is based on the observation that TCP source transmits new segments when it receives ACKs from the destination. Therefore, by shaping the TCP ACK stream, one can control the transmission rate at the TCP source. If the ACK stream shaping is accomplished somewhere between the destination and the source, neither the destination, nor the source needs to be aware of this, and hence it becomes an end-system transparent solution.

ACK-bucket itself is a simple concept borrowed from the well-known *leaky bucket* scheme used in ATM networks to control the rate of cells

entering the network from a given VC. The concept of ACK-bucket is illustrated in Figure 9.20. TCP ACKs arrive to the bucket according to TCP dynamics and drain out at a controlled rate so that the TCP transmission rate matches the ABR to eliminate congestion at IP-ATM gateway.

9.6.2 ACK-bucket implementation at gateway

To control the congestion at the IP-ATM gateway (in the forward path), the most logical place to implement the ACK-bucket control is in the gateway itself (in the reverse path). Figure 9.21 shows the location of an

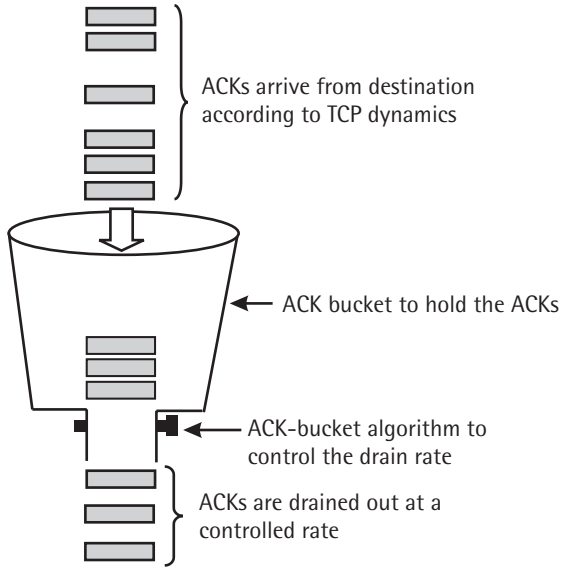


Figure 9.20 ACK-bucket operation.

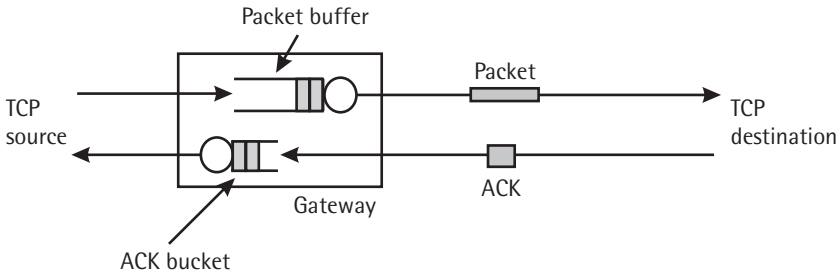


Figure 9.21 ACK-bucket control in an IP-ATM gateway.

ACK-bucket in an IP-ATM gateway. At first it may seem that the reduction in packet buffer is achieved at the expense of ACK buffer. In fact, because ACKs are much smaller than original data packets, the overall (packet buffer + ACK buffer) buffering requirement in the gateway is significantly reduced.

When the ACKs are independent TCP segments (carrying no data in the reverse path), they can be held and delayed in the bucket without causing any delay to reverse traffic. However, if ACKs are piggybacked on data packets on the reverse path (e.g., for Telnet echo packets carrying ACKs), the acknowledgment number needs to be extracted from the header and stored in the bucket; the header is modified so that it does not carry new acknowledgment, and the TCP data segment is released without holding it. When the ACK needs to be released, a new TCP ACK segment is created and released.

9.6.3 ACK-bucket algorithms

One of the main tasks of an ACK-bucket is to schedule the release of ACKs from the bucket in such a way so that the data arriving rate to the gateway from TCP sources matches the allowed departure rate at the ABR link. Although it may seem a trivial task, the ACK scheduling becomes quite complicated when sources implement their own congestion-control algorithms and react to an ACK differently at different times depending on the system state. This is particularly true for TCP sources, which implement complex end-to-end flow and congestion-control algorithms as discussed in Chapter 2.

There have been a few contributions to the ATM Forum proposing new ACK-releasing algorithms. The following subsections describe several proposed algorithms to control the release of ACKs from the ACK-bucket in the gateway.

9.6.3.1 TCP source modeling

As discussed in Chapter 2, TCP source transmission is regulated by a congestion window. Upon reception of an ACK, the TCP source will transmit either one or two segments depending on the congestion window size at that time. To accurately predict the source response to receiving an ACK, the gateway must be able to keep track of the window dynamics at the TCP source. Narvaez and Siu [14] originally proposed an algorithm that models the source behavior at the gateway to keep track of the window dynamics at

the source. Using the source model, the gateway can make effective decisions regarding when to release the next ACK for a given ABR bandwidth.

The strength of TCP source modeling is its ability to *exactly* predict the amount of data that will be transmitted by the source when an ACK is released. This exact prediction provides a tight control over the traffic flow into the gateway and can practically eliminate (in the steady state) the buffering requirement at the gateway in the forward path.

The main drawbacks of TCP source modeling are (1) that the gateway needs to know the TCP version of the source and (2) that it (for each TCP flow) significantly increases implementation complexity at the gateway.

9.6.3.2 Queue threshold

To address the complexity problem, Narvaez and Siu [16] later modified their TCP source-modeling algorithm and proposed a queue threshold-based algorithm similar to the one used in the HGRCP scheme described in Section 9.5. This scheme tries to keep the *effective* queue size close to, but smaller than, a predetermined queue threshold in the forward buffer. Here the effective queue size is the summation of current queue size plus an estimate of the amount of data that is in transit from the source to the gateway. An ACK is only released from the ACK-bucket as long as the estimated effective queue size is smaller than the queue threshold.

The effective queue size can be estimated without modeling the source TCP, although such an estimation will be inexact. If the source is modeled, the estimate becomes an exact estimation. Thus there is a tradeoff between the complexity and accuracy of the prediction (and hence the buffering requirement) using the queue threshold algorithm in [16].

9.6.3.3 ACR and queue threshold

Another algorithm proposed in a recent ATM Forum contribution [15] uses the current ACR value in conjunction with a predetermined queue threshold Q_{thres} to schedule the ACK release from the ACK-bucket. This algorithm uses two parameters, L and H, to control the ACK release. When the current queue size exceeds Q_{thres}, ACK is released at the rate ACR/H; otherwise the ACK is released at ACR/L.

9.6.3.4 ERICA+

ERICA+ is a well-known ER calculation algorithm for ABR switches. In another recent contribution to the ATM Forum, Satyavolu et al [17]

proposed using a simplified ERICA+ algorithm at the IP-ATM gateway to compute the ER for each contending TCP connection and use this ER to regulate the ACK from the corresponding ACK-bucket.

9.6.4 Simulation of TCP rate control

Performance of TCP rate control with the ACR and queue threshold algorithm (Section 9.6.3.3) was evaluated [15] over a long-distance ABR link using the simulation configuration shown in Figure 9.22. In this configuration, each router acts as an IP-ATM edge device and implements the ACK-bucket scheme. Table 9.6 lists the values of some key parameters used in the simulation.

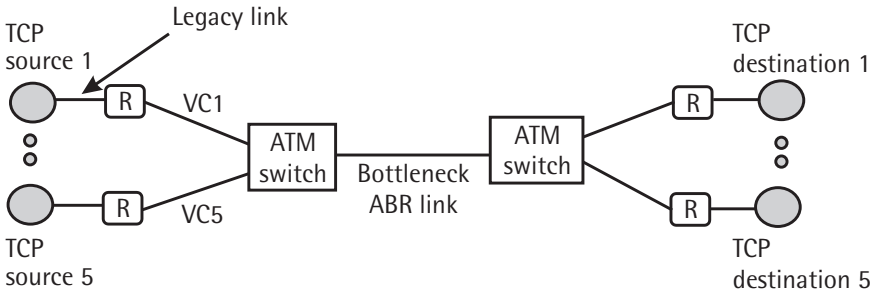


Figure 9.22 Simulation configuration for TCP rate control.

Table 9.6
Simulation Parameters for TCP Rate Control

Simulation Parameter	Value
MCR of each VC	0 Mbps
Physical rate of ATM backbone link	150 Mbps
TCP window size	128 KB
Router forward packet buffer size	10 packets
Packet Size	9,188 bytes
Queue threshold (Q_{thres})	3 packets
H	2.08
L	0.5
Source-router and router-switch distances	1 Km

Several simulations were run, each with a different speed for the legacy links between the sources and the routers. One set of simulations were run with TCP rate control implemented and another without the TCP rate control. For all link speeds, higher TCP throughputs were obtained for all sources with TCP rate control implemented at the routers. TCP throughputs, with and without TCP rate control, for Source 1 are shown in Table 9.7 for different legacy link rates. It can be seen that in this simulation, this source was able to achieve 20–40% performance improvement with TCP rate control for the range of legacy link rates considered.

9.6.5 Commercial products

Packeteer Inc., a high-tech start-up, recently released a product called PacketShaper™ [18] to manage congestion at the LAN-to-WAN link by shaping the TCP ACK stream in the LAN-to-WAN gateway. Figure 9.23

Table 9.7
TCP Throughput With and Without TCP Rate Control

Legacy Link Rate (Mbps)	Without TRC (Mbps)	With TRC (Mbps)
50	6.5	8.1
100	9.0	16.1
150	14.0	24

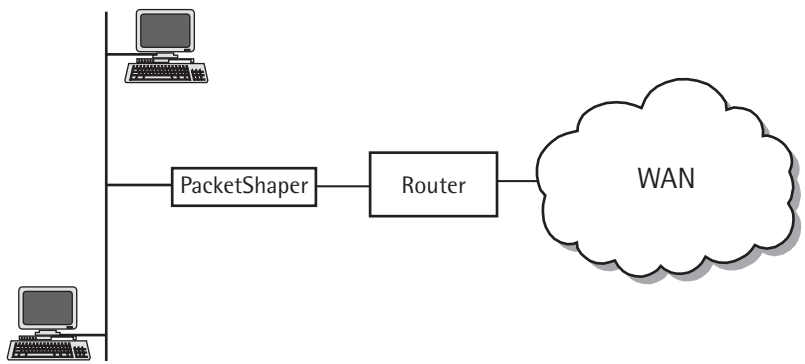


Figure 9.23 TCP ACK shaping using PacketShaper.

illustrates the location in the campus network where the PacketShaper is deployed. Because it sits between the gateway and the rest of the campus hosts, it has the same access to forward and backward traffic as the gateway. Therefore, a PacketShaper can do the TCP ACK shaping without interfering with the existing gateway. Because PacketShaper is based on layer 4 rate control, it can allow the network manager to allocate arbitrary rates or bandwidth to specific applications.

PacketShaper is primarily designed to work with leased lines such as T1 and T3 WAN links where the WAN link bandwidth remains constant 24 hours a day, seven days a week. Because PacketShaper does not make use of ABR bandwidth parameters (it sits "behind" the edge device and does not have access to ABR parameters), such as ER and ACR feedback from the ATM network, it is likely to be less effective in controlling the congestion at IP-ATM edge devices.

9.6.6 ACK-bucket limitations

Although ACK-bucket provides a transparent solution to the gateway congestion problem without requiring any changes to TCP or IP, it is not without its limitations. A serious limitation of the ACK-bucket scheme is that it needs access to the TCP header, for both reading and writing. Since the TCP header is encapsulated in the IP payload, the ACK-bucket scheme, therefore, needs access to the payload of each IP packet entering into the campus network. Consequently, ACK-bucket cannot be implemented easily in virtual private networks (VPNs) where the IP payload is encrypted for security reasons.

Another inherent limitation of ACK-bucket is its reliance on TCP or TCP-like "closed-loop" protocols that receive ACKs from the receiver. The scheme cannot work with UDP or other "open-loop" protocols that do not rely on ACKs from the receiver. ACK-bucket, therefore, is not very effective in controlling congestion in the IP-ATM gateway when there is a significant amount of UDP traffic.

9.7 Comparison of schemes

This section qualitatively compares various end-to-end traffic management schemes for IP/ATM internetworks. Table 9.8 compares the schemes based on their congestion notification methods, level of feedback

Table 9.8
Comparison of Various End-to-End Traffic Management Schemes for IP/ATM Internetworks

Scheme	Congestion Notification	Level of Feedback	TCP-Dependent	TCP/IP Modification	Access to IP Payload	Implementation
Adaptive packet discard [1]	Implicit	Binary	Yes	No modification	No	Not implemented
ECN (RFC 2481)	EFCN	Binary	Yes	Modification at both TCP and IP layers	No	Implemented on Linux, FreeBSD
Salim et al. [10]	EBCN	Multilevel	Yes	Modification at both TCP and IP layers	No	Not implemented
HGRCP [11]	EBCN	ER	No	Modification at IP layer	No	Implemented on Linux platform
TCP rate control	Implicit	ER	Yes	No modification	Yes	Commercial product

provided, dependence on TCP or TCP-like protocols, changes required for TCP/IP header or protocol, and access to IP payload. The last column of Table 9.8 indicates whether the scheme has been implemented and tested in an experimental or production environment.

Bagal et al [19] carried out a simulation study to compare the performance of ECN and TCP rate control using a simple network configuration of a single bottleneck link (similar to the one shown in Figure 9.4). Many combinations of various parameter values were explored. The results did not show any significant performance difference between ECN and TCP rate control as far as TCP throughput and router queue size are concerned. TCP rate control, however, performed better than ECN in terms of allocating the bottleneck bandwidth more fairly among the contending TCP sources.

No study has been reported in the literature comparing the performance of ECN or TCP rate control against HGRCP or other schemes shown in Table 9.8. Given that all proposed schemes were shown to improve TCP throughput and network delay for IP/ATM internetworks or even pure IP networks with congested bottleneck links, it will be the implementation features, such as whether the access to IP payload or changes to TCP/IP is necessary, that will drive the adoption of one scheme over the other.

9.8 Summary

Although ABR control is very effective in eliminating congestion inside the ATM network, in the IP/ATM internetworking environment, it merely shifts the congestion from inside the ATM network to the IP-ATM edge device. It is necessary to implement some additional control mechanism between the IP sources and the IP-ATM edge device to achieve effective end-to-end traffic management in IP/ATM internetworks.

Several end-to-end traffic management schemes have recently been proposed. ECN (RFC 2481) is a proposed EFCN method for TCP/IP networks that is currently being discussed in the IP community. The main idea behind ECN is to provide congestion feedback to TCP via IP packet headers without dropping IP packets at the gateway. As packet drops are minimized, packet retransmissions are also minimized, which, in turn, reduces packet delay and increases the performance of delay-sensitive applications. The main limitation of ECN is that it requires changes to TCP/IP and hence cannot be deployed transparently in existing networks.

TCP rate control, a scheme being discussed in the ATM Forum, is a transparent solution to end-to-end congestion control that controls the TCP source rate by shaping the ACK stream to the source. Although TCP rate control can be deployed immediately without changing existing TCP/IP software, its main limitation lies in the fact that it requires access to the IP payload to work with the TCP ACKs. TCP rate control, therefore, cannot be implemented easily in VPNs where the IP payloads are encrypted for security reasons.

HGRCP is a layer 3 (IP layer) rate control mechanism between the IP-ATM edge device and the LAN hosts. Because it works only at layer 3, it does not need access to the IP payload and hence can be readily implemented in VPNs. The main limitation of HGRCP is that it requires changes to the IP layer and hence (like ECN) cannot be implemented transparently (without modifications of IP software) in existing networks.

ECN, TCP rate control, and HGRCP [20] are all effective in addressing the congestion problem at the IP-ATM edge device. It should be noted, however, that no end-to-end traffic management scheme has been standardized yet by the IETF or the ATM Forum.

References

[1] Jagannath, S., and N. Yin, "End-to-End Traffic Management Issues in IP/ATM Internetworks," *Internet Draft: draft-jagan-e2e-traf-mgmt-00.txt,* August 1997.

[2] Floyd, S., "TCP and Explicit Congestion Notification," *Computer Communication Review*, Vol. 24, No. 5, pp. 8–23, October 1994.

[3] Ramakrisnan, K., and S. Floyd, "A Proposal to Add Explicit Congestion Notification (ECN) to IP," *Internet RFC 2481*, IETF, January 1999.

[4] Postel, J., "Internet Protocol," *Internet RFC 791,* IETF, September 1981.

[5] Deering, S., and R. Hinden, "Internet Protocol, Version 6 (Ipv6) Specification," *Internet RFC 2460*, IETF, December 1998.

[6] Nichols, K., et al, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," *Internet RFC 2474*, IETF, December 1998.

[7] Krishnan, H., "Analyzing Explicit Congestion Notification (ECN) Benefits for TCP," Masters Thesis, University of California at Los Angeles, 1998.

[8] "The ECN Web Page," URL http://www-nrg.ee.lbl.gov/floyd/ecn.html.

[9] Saito, H., et al., "Performance Issues in Public ABR Service," *IEEE Communications Magazine*, Vol. 34, No. 11, 1996, pp. 40–48.

[10] Salim, H. J., et al., "A Proposal for Backward ECN for the Internet Protocol (IPv4/IPv6)," *Internet Draft: draft-salim-jhsbnns-ecn-00.txt,* June 1998.

[11] Hassan, M., J. W. Breen, and M. Atiquzzaman, "HGRCP: A Rate-Based Flow Control Mechanism for Intranets Interconnected by ATM Networks," *Computer Networks,* Vol. 31, No. 22, December 1999, pp. 2359–2377.

[12] Hassan, M., H. Sirisena, and M. Atiquzzaman, "A Congestion Control Mechanism for Enterprise Network Traffic over ATM Networks," *Computer Communications,* Vol. 22, No. 14, September 1999, pp. 1296–1306.

[13] Biggs, M., M. Hassan, and J. W. Breen, "Implementation and Experimentation of IP Rate Control," *Proc. of IEEE International Conference on Networks (ICON '99),* Brisbane, September 1999, pp. 242–249.

[14] Narvaez, P., and K. Y. Siu, "An Acknowledgment Bucket Scheme for Regulating TCP Flow over ATM," *Computer Networks,* Vol. 30, No. 19, October 1998, pp. 1775–1791.

[15] Koike, A., "TCP Flow Control With ACR Information," ATM Forum Contribution, 97-0758R1, December 1997.

[16] Narvaez, P., and K. Y. Siu, "New Techniques for Regulating TCP Flow over Heterogeneous Networks," *Proc. IEEE Conference on Local Computer Networks,* Boston, October 1998.

[17] Satyavolu R., K. Duvedi, and S. Kalyanaraman, "Explicit Rate Control of TCP Applications," ATM Forum Contribution, 98-0152R1, February 1998.

[18] "PacketShaper Data Sheet," Packeteer, Inc. Web site: http://www.packeteer.com.

[19] Bagal, P., S. Kalyanaraman, and B. Packer, "Comparative Study of RED, ECN and TCP Rate Control," Technical Report, Department of ECSE, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, March 1999.

[20] The HGRCP Web page, http://www.sd.monash.edu/~mahbub/HGRCP/hgrcp.htm.

10

TCP Deadlock

10.1 Introduction

As mentioned in Section 5.6, TCP connections over an ATM network may enter into a cyclic wait phenomenon: The TCP cycles through the transmit state followed by the wait state, followed by the transmit state, and so on. This stop-and-start behavior dramatically reduces TCP throughput. TCP enters the wait state when the sender and the receiver enter a deadlock—the sender is waiting for an ACK from the receiver before it can send more data, and the receiver is waiting for more data from the sender to generate an ACK. TCP comes out of the wait state when an ACK timer expires at the receiver causing the receiver to send an ACK without waiting further. Figure 5.5 illustrates the cyclic wait phenomenon caused by TCP deadlocks.

The TCP deadlock phenomenon over ATM networks was discovered in 1995 independently by Comer and Lin [1] and Moldeklev and Gunningberg [2]. After detailed investigations, these researchers identified several factors, some inherent to the TCP protocol and others originating from software implementation of the TCP protocol, which contributes to such

deadlocks. They also proposed measures to prevent TCP deadlock. This chapter explains the causes of TCP deadlock, the impact of deadlock on TCP performance, and measures to prevent TCP deadlock.

10.2 Causes of deadlock

TCP deadlock is the result of interactions among several factors. Some of these factors are TCP protocol–specific, while others are implementation-dependent. This section describes each of the factors contributing to TCP deadlock. The following sections detail how these factors cause deadlock.

This section describes TCP implementation in SunOS 4.1.1 as an example implementation, as this implementation was used in the experiments [1, 2] to detect the TCP deadlock problem for TCP over ATM. (SunOS 4.1.1 TCP is a version of 4.3 BSD-Tahoe TCP [3].) Since other TCP implementations, especially UNIX-based implementations, use techniques similar to those implemented in SunOS, it is likely that the deadlock situations explained in this section for SunOS will also occur in other platforms. Indeed, many researchers have reported [2] a performance bottleneck for TCP over ATM for SPARC/Solaris, SGI/IRIX, and IBM RS6000/AIX. However, it has not been established whether the performance bottleneck for these platforms is actually due to TCP deadlock. The lack of free source code for these commercial platforms makes it difficult to analyze the source of the performance problem.

10.2.1 Combination of send-and-receive socket buffers

TCP is usually implemented (especially in UNIX systems) in the kernel of the host operating system. Applications communicate with the TCP via the socket layer, which lies between the application layer and the kernel. The socket layer provides an application programming interface (API), such as *write* and *read* system calls and socket buffers, for applications trying to communicate with the protocols in the kernel.

A socket identifier is used with each *write* and *read* system call to uniquely identify a given application process. Associated with each socket identifier are two buffers—one for copying data from application to kernel using a *write* call (send buffer), and one for moving data from kernel to application using a *read* call (receive buffer). The user can select the size of send-and-receive socket buffers.

Since the TCP sender and receiver are two separate application processes usually running on two different hosts across a network, it is not unlikely that the socket buffer sizes at the sender (send buffer size) and receiver (receiver buffer size) may be different. Most combinations of send and receive buffer sizes are harmless (deadlock-free). However, there are combinations in which the send buffer size is smaller than the receive buffer size that may lead to TCP deadlock. These combinations and their effect will be discussed in Section 10.3.

10.2.2 Preventing small packet transmission at the sender (Nagle's algorithm)

Transferring small segments consumes unnecessary network bandwidth and creates unnecessary processing load on the sender and receiver. The waste of network bandwidth is due to high packet overhead to data ratio for small packets, and the increased processing load is caused by creating and processing too many packets and their acknowledgments.

To prevent the sending TCP from sending small segments, Nagle [4] proposed a simple algorithm that is very easy to implement. Nagle's algorithm can be summarized as follows:

> If previously transmitted data remains unacknowledged, the sending TCP should refrain from sending segments smaller than MSS and must wait until there is enough data to fill a maximum-size segment or until all the outstanding (unacknowledged) data has been acknowledged.

Therefore, with Nagle's algorithm implemented, the sender may have to wait for acknowledgment from the receiver to send more data even if there is data waiting in the send buffer ready to be transmitted. Although the user has the option of switching off Nagle's algorithm, it is the default and recommended option for many TCP applications, including Telnet and FTP.

10.2.3 Network MTU

Nagle's algorithm tries to prevent transmission of segments smaller than MSS. TCP MSS depends directly on the size of MTU—MSS is MTU minus the TCP and IP headers—of the underlying network. Therefore, the MSS for TCP over ATM will be much larger than the MSS over traditional networks, as IP over ATM has a much larger MTU than conventional

networks (see Table 5.3). Therefore, with Nagle's algorithm implemented, TCP over ATM has a higher probability of entering a wait situation where the TCP sender has to wait for acknowledgment from the receiver to send more data even though there is data waiting to be sent in the send buffer. Details of MSS's role in causing TCP deadlock will be explained later in Section 10.3.

10.2.4 Delaying acknowledgment at receiver

The acknowledgment strategy [5] mandates that the receiving TCP should delay ACKs (i.e., instead of generating an ACK immediately upon receiving a segment, the receiver must wait a while) to minimize processing and traffic overhead. By delaying an ACK, the receiving TCP avoids sending one ACK per segment received—that is, if more segments arrived during the delay period, a single ACK segment can acknowledge all these segments. For full-duplex communication (the receiving TCP also sends data segments to the sending TCP, such as ECHO packets in a TELNET session), a delay may permit the acknowledgment to be piggybacked on a data segment. In some cases, the delay permits the receiving TCP to send a single segment that acknowledges data and advertises the current receiver's window.

How much delay is acceptable? A rule of thumb is that an ACK should not be delayed excessively, as it may cause the retransmit timer at the sender to expire. Moreover, the sender uses the received ACKs to estimate the round-trip time of an end-to-end connection. To prevent excessive delays, RFC 1122 [6] makes the following recommendations:

- There should be at least one ACK for every two segments received. In other words, with delayed ACK option, an ACK should be generated when *two* segments are received.
- The above recommendation may not prevent excessive delays if the receiver is very slow and the application is reading data from the buffer very slowly (data cannot be acknowledged unless the application reads it successfully from the buffer). As a measure against slow receivers, RFC 1122 [6] specifies that an ACK should not be delayed more than 500 ms.

To address the above recommendations, SunOS 4.1.1 implements the following:

- Generate an ACK if data received is the equivalent of two MSSs. However, for small receive buffers (compared to the MSS), a single segment alone may significantly reduce the available buffer space, and unless an ACK is generated immediately, the sender may have to wait for acknowledgment (as its window will be full soon). To allow the sending TCP to send more data, SunOS generates an ACK immediately (without waiting for receiving two segments) when received data is at least 35% of receiver buffer size.
- Schedule a delayed ACK timer every 200 ms; when the timer expires, transmit an ACK if ACKs have been delayed.

Therefore with SunOS, an ACK may be generated either when "enough" data is received "quickly" or when an ACK delay timer expires. If there are too many timer-generated ACKs, TCP throughput will be reduced, since the sender would be waiting for ACKs.

10.2.5 Sender action sequence on acknowledgment reception

The generation of new segments at the sender on receiving acknowledgment from the receiver is one of the factors that causes deadlocks, and it is implementation-dependent. On receiving an acknowledgment, some buffer space is freed in the send buffer. Therefore, it would be possible to first copy more data from an application into the send buffer and then send a larger segment. Instead, in most UNIX systems, the sequence of action is to send the available data in the send buffer before copying more data into the buffer. This is implemented to avoid the delay involved in waking up an application (which was put to *sleep* due to lack of space in the send buffer).

As a consequence of the above sequence of actions, a TCP sender may send segments smaller than MSS. As will be shown later, this order of actions leads to TCP deadlock and has a devastating effect on TCP throughput.

10.2.6 Adding data to the TCP send buffer

The application wishing to transmit data using TCP uses the *write* system call and passes the data (actually the pointer to the start of data in memory and the size of data) as a parameter to the function. The application data, passed as a parameter to the write system function, is copied and added to the TCP send buffer according to the following rules.

- SunOS imposes a 4K limit on the number of bytes copied to the socket send buffer before the TCP transmit routine is called. According to this 4K limit, if the send buffer has space larger than 4K (4,096) octets, and the application has more than 4K data, SunOS adds user data to the TCP send buffer in blocks of 4K and invokes the TCP transmit routine after each block of 4K data is added. If the available space in the TCP send buffer is smaller than 4K, SunOS adds data in multiples of 1K. If the application has less than 4K data, all data are copied before TCP transmit routine is called. The original reason for copying 4K data was that most traditional network MTUs were smaller than, or close to, 4K.

It should be mentioned that in SunOS, the data from send buffer is not removed immediately (in case they need to be retransmitted) after transmission. Such data remains in the send buffer until acknowledgment by the receiver. This further limits the available space in the send buffer.

10.3 TCP deadlocks

TCP deadlocks depend significantly on the combinations of socket buffer sizes at the sender (send buffer) and receiver (receive buffer). Not all combinations cause TCP deadlock. This section identifies the harmful combinations (i.e., the combinations that cause TCP deadlock) and explains with examples how deadlock can occur with these combinations of send-and-receive buffer sizes.

10.3.1 Small send buffer

Small send buffers are more likely to cause TCP deadlocks. As explained in Section 10.2.4, the receiver refrains from (delays) sending ACK until it receives data equal to or greater than either 35% of the receive buffer size or 2 MSS. Therefore, TCP deadlock can be easily predicted for the following Boolean expression

$$\left(S < 0.35R\right) \text{ and } \left(S < 2 \text{ MSS}\right) \tag{10.1}$$

where S is the send buffer size, and R is the receive buffer size. When (10.1) is true, there is no way the sender can send a segment large enough to cause

immediate acknowledgment at the receiver. Because the MSS the sender can send is smaller than the required size to generate immediate acknowledgment, the acknowledgment will always be delayed and eventually transmitted at the expiry of the timer. Example 10.1 shows how TCP enters deadlock for a combination of send-and-receive buffer sizes that satisfy (10.1).

Example 10.1

Consider a send-and-receive buffer combination of S = 8K and R = 24K. With this combination, S = 0.33R and is smaller than 2 MSS for TCP over ATM. (The default MTU for TCP/IP over ATM is 9,180 bytes [7], which yields an MSS = 9,140 bytes for the standard 20 bytes of TCP header and 20 bytes of IP header.) According to the data-copying rule (Section 10.2.6), the sender copies 4K user data into the send buffer and calls the TCP transmit routine. When these 4K data arrive at the receiver, the receiver does not generate an ACK immediately as the data received is smaller than 2 MSS and smaller than 0.35R (due to the ACK delay rules explained in Section 10.2.4).

After transmitting the first 4K data, the sender copies another 4K data into the send buffer. However, because there is data outstanding in the send buffer waiting for acknowledgment, the sender cannot transmit the next 4K data. (Nagle's algorithm as explained in Section 10.2.2.) The send buffer is now full, and there is no space in the send buffer to copy more data. Now the receiver is waiting for more data from the sender before it can send an ACK, and the sender is waiting for an ACK from the receiver before it can transmit more data. The connection, therefore, enters a deadlock. The status of send-and-receive buffers when the deadlock is caused is shown in Figure 10.1.

The deadlock can only be broken when the ACK delay timer (a timer is set every 200 ms in SunOS) expires at the receiver, and an ACK is generated. (This is a timer-generated ACK.) However, when the ACK for the first

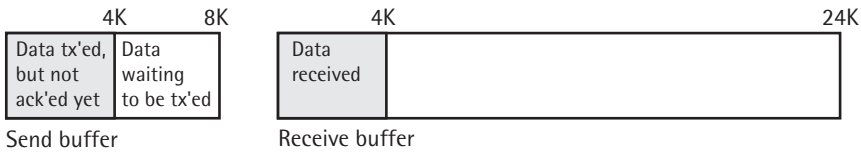


Figure 10.1 Send-and-receive buffer status when the connection enters deadlock for the first time for $S = 8K$ and $R = 24K$.

4K data is received at the sender, the sender first sends the second 4K data waiting in the buffer, copies another new 4K data into the buffer, and goes straight into deadlock once again. The cyclic deadlock phenomenon continues with an effective throughput of 4K every 200 ms.

Note that with combinations of socket buffer satisfying (10.1), the TCP connection will enter deadlock independent of Nagle's algorithm (Section 10.2.2.), the data-adding rule to the send buffer (Section 10.2.6), and sender action sequence on acknowledgment reception (Section 10.2.5). This is because even if the sender was allowed to transmit a buffer full of data at the same time, the receiver would still not be able to generate an ACK immediately, as the send buffer size is too small in this case.

10.3.2 Implementation-dependent deadlocks

At first it may seem that combinations of send-and-receive buffers that do not satisfy (10.1) are free from deadlocks. However, careful investigation [2] revealed that due to action sequence on ACK reception (Section 10.2.5) and data-adding rules to the send buffer (Section 10.2.6)—both of which are implementation-specific in many popular UNIX systems—deadlock can still happen for combinations that do not satisfy (10.1) but satisfy the following Boolean expression

$$\left(S < 0.35R + \text{MSS}\right) \text{ and } \left(S < 3\ \text{MSS}\right) \tag{10.2}$$

Example 10.2 shows how deadlock can still happen in such situations.

Example 10.2

Let's consider the combination S = 8K and R = 16K. In this case, the send buffer is 50% of the receive buffer and hence does not fall in the range of (10.1), but for MSS = 9,140, it satisfies (10.2). Initially, 4K is copied into send buffer and transmitted before another 4K is copied. The receiver cannot generate an ACK for 4K, as it is smaller than 2 MSS and 0.35R. The sender cannot send more data, as there is unacknowledged data in the send buffer, and there is not enough space in the send buffer to create a MSS segment. The connection deadlocks and the deadlock is broken by the timer-generated ACK at the receiver.

When the ACK for the first 4K is received at the sender, the sender first sends the remaining 4K before copying another 4K (according to the action

sequence explained in Section 10.2.5) and hence goes into another deadlock. Note that, for this example, the deadlock would not occur if the action sequence allowed the duplication of another 4K before transmitting the waiting 4K in the send buffer; this would allow the sender to transmit an 8K segment (50% of receive buffer), which would cause the receiver to generate an ACK immediately. Similarly, if the data-adding rule (Section 10.2.6) allowed a calling of the TCP transmit routine after copying 8K data into the send buffer, the deadlock could be avoided.

10.3.3 Deadlock zones

It should be clear by now that deadlock is caused only for certain combinations of send-and-receive buffer sizes; other combinations are safe and do not cause deadlocks. Table 10.1 summarizes the combinations for which TCP deadlock can occur.

Figure 10.2 shows the deadlock zones in the square of all possible combinations of send-and-receive buffer sizes for a maximum of 64K (unless the window scale option is used, the maximum TCP window size is 64K) send and receive buffer sizes for an MSS of 9,140 octets. In the dark gray zone, the deadlock will occur independent of implementation (first row in Table 10.1), and deadlock will happen in the light gray zone if the implementation follows the data copy rules explained in Sections 10.2.5 and 10.2.6 (second row in Table 10.1). The white zone is deadlock-free. Figure 10.2 clearly shows that TCP deadlocks may occur only for certain combinations of send-and-receive buffer sizes when the send buffer is smaller than the receive buffer.

Table 10.1
Send-and-Receive Socket Buffer Size Combinations
That May Cause TCP Deadlock

Send-Receive Buffer Combination	Type of Deadlock
($S < 0.35R$) and ($S < 2$ MSS)	Deadlock will occur in all platforms; implementation-independent
(($S > 0.35R$) or ($S > 2$ MSS)) and (($S < 0.35R + \text{MSS}$) and ($S < 3$ MSS))	Deadlock will occur in SunOS and other platforms, implementing data-copying rules explained in Sections 10.2.5 and 10.2.6

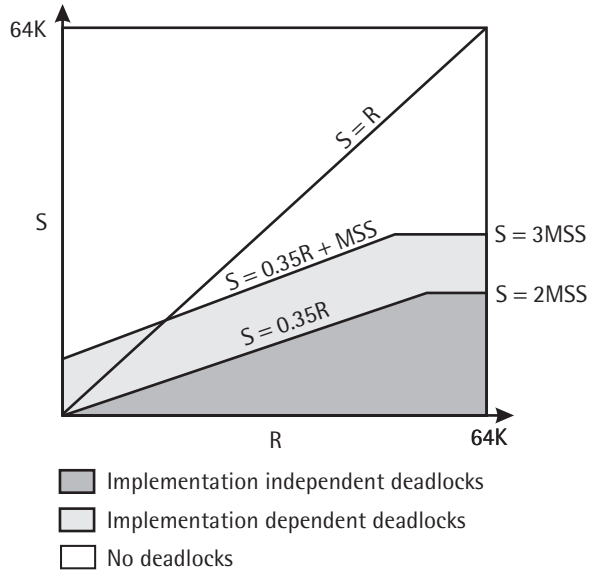


Figure 10.2 Deadlock zones for TCP over ATM with MSS = 9,140 bytes.

10.4 Role of MSS and MTU

TCP MSS, which is directly related to the MTU of the underlying physical network, plays a significant role in the area covered by the deadlock zones. The larger the MSS, the greater the deadlock zone and the greater the possibility of deadlock. For MSS = 9,140 (for default MTU = 9,180 bytes for IP over ATM [6]), the deadlock zone covers almost 25% of the total area of possible send-and-receive buffer size combinations (see Figure 10.2). However, it can be seen in Figure 10.3 that for TCP over Ethernet (Ethernet has an MTU = 1,500 bytes causing a TCP MSS = 1,460), the deadlock zone is less than 10% of the total send-and-receive buffer combination area.

10.5 Impact of deadlocks on TCP throughput

To assess the impact of deadlock on TCP performance, Moldeklev and Gunningberg [2] conducted an experiment with two Sparc10 machines (one a TCP sender and the other a TCP receiver) connected via a FORE

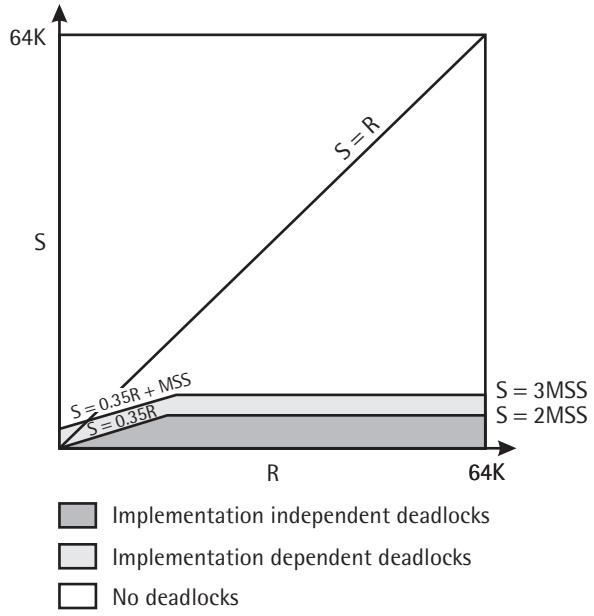


Figure 10.3 Deadlock zone for TCP over Ethernet with MSS = 1,460 bytes.

ATM switch, as shown in Figure 10.4. The Sparc10 machines were running SunOS 4.1.3. The default MTU for the FORE ASX-100 ATM switch was 9,188. Many tests were conducted for different combinations of TCP send-and-receive buffers.

Table 10.2 presents TCP throughputs obtained for various combinations of send-and-receive buffer sizes. Dark-shaded entries correspond to implementation-independent deadlocks due to "small" send buffers, as explained in Section 10.3.1. Light-shaded entries correspond to deadlocks

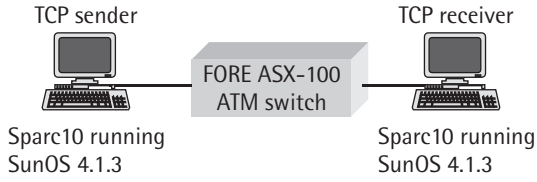


Figure 10.4 Configuration of a TCP/ATM network to measure TCP throughput.

Table 10.2
TCP Throughput (in Mbps) for Different Combinations of Send (S)-and-Receive (R) Buffer Sizes

S\R	4K	8K	16K	24K	32K	40K	48K	52K	56K	64K
4K	23	23	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
8K	26	32	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
16K	26	35	38	0.3	0.3	0.4	0.4	0.4	0.4	0.4
24K	26	34	40	40	40	40	0.8	2	4	3
32K	24	32	35	51	47	47	47	47	47	47
40K	26	35	36	55	58	58	58	57	58	56
48K	26	36	37	56	58	62	63	63	63	64
52K	26	35	37	54	58	62	63	64	64	64
56K	26	35	37	56	61	63	64	64	64	64
64K	26	36	35	54	61	62	62	63	63	64

that are implementation-dependent (due to data copy rules for sending buffers as explained in Section 10.3.2).

As can be seen from Table 10.2, TCP deadlock has a detrimental effect on TCP throughput. For example, with $S = 8K$ and $R = 8K$ (no deadlock), 32 Mbps TCP throughput was achieved. However, when the receive buffer size was increased to 16K and above (now deadlocks will occur due to a "small" send buffer size, as explained in Section 10.3.1), the throughput dropped dramatically to only 0.1 Mbps!

A quick glance through Table 10.2 confirms that TCP performance is severely degraded only for those combinations of send-and-receive buffer sizes as summarized in Table 10.1. This observation, in turn, confirms that the low throughput figures in Table 10.2 are due to TCP entering into circular deadlocks.

10.6 Preventing TCP deadlocks

There are several solutions to the deadlock problem. This section examines and discusses the merits and drawbacks of each solution.

10.6.1 Turning off the delayed ACK option

As explained in Section 10.2.4, a delayed acknowledgment policy at the receiver is one of the main factors causing TCP deadlock. If the receiver explicitly acknowledges every single segment it receives, the deadlock does not happen. In some operating systems, such as in SunOS 4.1.x, explicit acknowledgment of every segment is a kernel compile option. The tradeoff of this option is increased processing load and traffic overhead due to an increased number of ACKs.

10.6.2 Turning off Nagle's algorithm

It is due to Nagle's algorithm (see Section 10.2.2) that TCP enters a deadlock even when there is data in the send buffer ready to be transmitted. Therefore, one straightforward way to prevent deadlock is to turn off Nagle's algorithm. The user can turn off Nagle's algorithm (by setting TCP_NODELAY option) without recompiling the operating system. However, by turning off Nagle's algorithm, the user will incur the costs associated with small packet transmissions (see Section 10.2.2).

10.6.3 Send buffer larger than receive buffer

As Figures 10.2 and 10.3 show, there is no deadlock when the send buffer is equal to or larger than the receive buffer. For this solution, the sender needs to know the socket buffer size at the receiver. The sender will need to monitor the window update packets from the receiver to determine the buffer size at the receiver.

10.6.4 Send buffer no less than 3 MSS

Figures 10.2 and 10.3 clearly show that deadlock can be avoided by selecting a send buffer size that is larger than or equal to 3 MSS. This is the most elegant solution to preventing TCP deadlock, since it does not depend on receiver buffer size. Moreover, it works even with Nagle's algorithm and delayed ACK options. In other words, this solution will work with existing implementations of TCP in any platform.

Table 10.3 shows the send buffer sizes needed for this solution to prevent TCP deadlock for ATM and Ethernet networks. As can be seen, TCP/ATM requires more than five times as much send buffer as TCP/Ethernet to prevent TCP deadlock. Deadlock can be completely prevented in a TCP/ATM networking environment by selecting a send buffer size of

Table 10.3
Send Buffer Sizes ($S \geq 3$ MSS) to Completely Prevent TCP Deadlocks in ATM and Ethernet Networks

MSS	Send Buffer Size To Prevent Deadlock ($S \geq 3$ MSS)
9,140 (TCP/ATM)	27K
1,460 (TCP/Ethernet)	5K

27K or larger. TCP throughput data in Table 10.2 confirms that there is no TCP performance degradation for a send buffer size greater than 27K.

10.7 Summary

Deadlock significantly reduces TCP throughput over ATM networks. TCP deadlock is primarily caused by interactions between a delayed ACK policy at the receiver, Nagle's algorithm at the sender, and a large MSS for TCP over ATM. Fortunately, not all combinations of send-and-receive socket buffer sizes cause deadlock. TCP deadlocks can be completely avoided by simply selecting send buffers larger than or equal to three times the TCP MSS.

References

[1] Comer, D. E., and Lin, J. C., "TCP Buffering and Performance Over an ATM Network," *Internetworking: Research and Experience*, Vol. 6, 1995, pp. 1–13.

[2] Moldeklev, K., and Gunningberg P., "How a Large ATM MTU Causes Deadlocks in TCP Data Transfers," *IEEE/ACM Transactions on Networking*, Vol. 3, No. 4, 1995, pp. 409–422.

[3] Leffler, S. J., M. K. McKusick, M. J. Karels, and J. S. Quaterman, *The Design and Implemenation of the 4.3BSD Unix Operating System*, Reading, MA: Addison-Wesley, 1990.

[4] Nagle, J., "Congestion Control in TCP/IP Internetworks," *Internet RFC 896*, IETF, January 1984.

[5] Clark, D. D., "Window and Acknowledgment Strategy in TCP," *Internet RFC 813*, IETF, July 1988.

[6] Braden, R., "Requirements for Internet Hosts—Communication Layers," *Internet RFC 1122*, IETF, October 1989.

[7] Laubach, M., "Classical IP and ARP over ATM," *Internet RFC 2225,* IETF, April 1998.

11

TCP Performance Over Satellite ATM Networks

11.1 Introduction

Due to their global coverage and broadcast capabilities, satellite ATM networks have recently attracted the attention of network service providers to provide high-speed data services on an international scale. Several standards bodies have started working on various challenges and issues relating to the interworking of satellite and ATM technology. Because satellite links have some unique properties not found in fiber-based terrestrial ATM networks, it is important to study the performance dynamics of TCP over satellite networks in order to better design Internet backbones, corporate LAN interconnections, and other TCP/IP-based data services over satellite ATM networks. This chapter presents an overview of satellite ATM networks, the protocol architecture for running TCP over satellite ATM, various TCP enhancements and extensions that are likely to improve the performance of TCP over satellite ATM networks, and some simulation and experimental results for TCP performance over satellite ATM networks.

11.2 Satellite ATM networks

In this section, we describe the architecture and properties of satellite ATM networks, as well as the motivation behind their installation. We also describe the protocol stack for TCP over satellite ATM and the various types of satellites used.

11.2.1 Recent interests

With the rapid growth of the Internet and an increasing demand for being "connected" anywhere at anytime, satellites will play a vital role in building the backbone of the future information superhighway. Since ATM is the only standard networking technology designed from the ground up as a true multiservice network capable of transporting all three major services—data, voice, and video—it is expected that ATM will be used as a "front technology" to reap the benefits of satellite networks. There are several benefits and motivations for satellite ATM networks over the traditional fiber-based terrestrial ATM networks:

- *Global coverage:* It is possible to provide connectivity for the entire planet using satellites. Such global coverage using terrestrial links is too costly, if not impossible. Communication facilities at remote areas, where it is very difficult to locate cables and wires, can be easily set up using satellite-based wireless systems.
- *Broadcast and multicast communication:* Due to their inherent broadcast capability, satellite networks make it much easier to achieve multipoint-to-multipoint communications, an important characteristic of many emerging multimedia applications (e.g., videoconferencing).
- *Backup for terrestrial connections:* Satellite channels can be used as a backup for failed terrestrial links.
- *Broadband communication:* The recent release of the Ka-band (see Section 11.2.4 for more on frequency bands) frequencies (20/30 Ghz) opens the door for satellite-based broadband communications to support the demand for multimedia and high-speed data communication applications.
- *Distance-independent communication cost:* Unlike terrestrial communication services, the cost for providing satellite communication

services does not depend on distance. Because the same satellite infrastructure is in use, a connection across the street costs the same to service as a connection across the continent.

11.2.2 Network architecture

Figure 11.1 illustrates a possible architecture for satellite ATM networks. Two terrestrial ATM islands are connected via a constellation of two satellites. Each ATM island may include several TCP/IP hosts connected to the ATM network using some TCP/IP over ATM interworking solution as described in Chapter 4. The connectivity of an ATM island to the satellite is accomplished through a ground station equipped with an appropriate satellite dish, a satellite modem, and an ATM-satellite interworking unit (ASIU). In the sky, two satellites are connected via an intersatellite wireless link for seamless global connectivity between any two ground stations through satellite constellations.

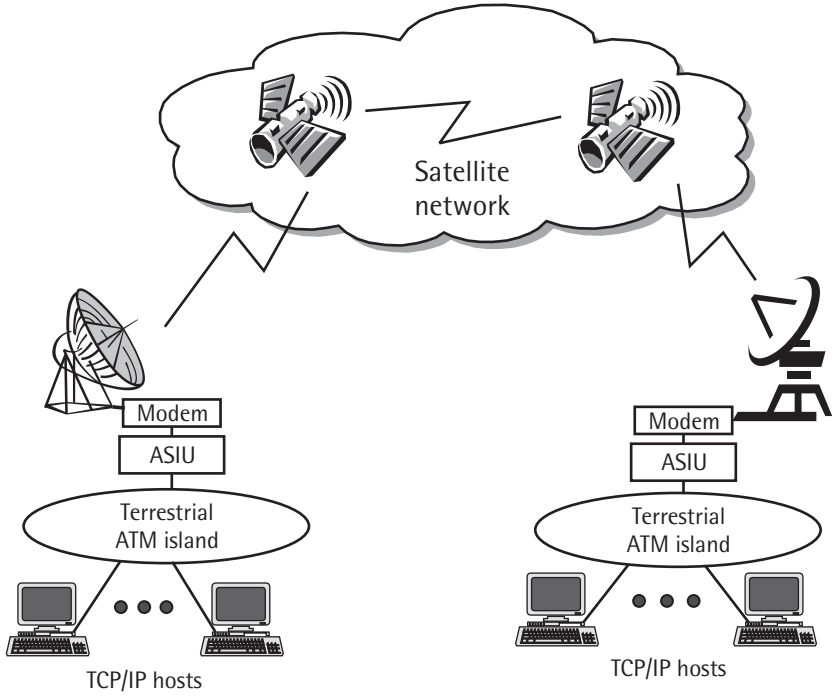


Figure 11.1 Satellite ATM network architecture.

11.2.3 Protocol stack for TCP over satellite ATM

Figure 11.2 shows the protocol stacks for running TCP/IP-based applications over satellite ATM networks when ATM-capable TCP/IP hosts are directly connected to an ASIU. Note that in this case, the TCP/IP hosts do not need to know whether the backbone connection is via satellite or terrestrial links. In Figure 11.2, the TCP/IP hosts implement the classical IP over ATM (see Chapter 4) solution for supporting TCP/IP over ATM.

11.2.4 Frequency bands

Satellites can operate in different frequency bands and use different carrier frequencies for the downlink (satellite-to-ground) and uplink (ground-to-satellite). Table 11.1 shows the three main commercial bands, their applications, their level of utilization, and their associated problems. The C band, operating at 4/6 GHz, was the first band designed for commercial use. Many satellites were launched operating in this band. Also, many telecommunication service providers use this band for their terrestrial microwave links. The C band, therefore, is already crowded. It is mainly used by broadcast TV. The benefit of the C band is a lower susceptibility to rain due to its low frequencies.

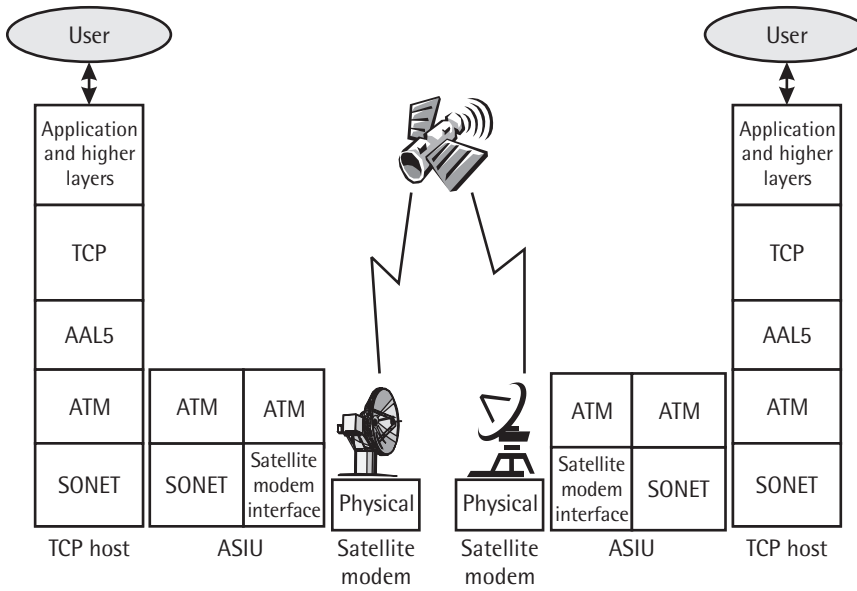


Figure 11.2 Protocol stacks for TCP/IP over satellite ATM networks [1].

Table 11.1
Commercial Frequency Bands for Satellite Communication

Band	Up/Down (GHz)	Applications	Problems
C	4/6	TV broadcast	Terrestrial interference
Ku	11/14	DirectTV	Rain
Ka	20/30	Broadband communications	Rain; high equipment cost

The next higher band designed was the Ku band operating at 11/14 Ghz. This band is used for DirectTV. Due to its higher frequency, this band suffers more from rain. The Ku band is also becoming saturated as more and more satellites use it. The recent trend is to utilize the Ka band at 20/30 Ghz. With a large frequency range, these are the next-generation satellites that will provide the satellite-based infrastructure for broadband, multimedia communication on a global scale. However, this high frequency also means a higher equipment cost and more rain attenuation.

In addition to the above commercial bands, there are some military bands that cannot be used for commercial purposes.

11.2.5 Geostationary earth orbit (GEO) and low earth orbit (LEO) satellites

Satellites rotate around the Earth on a predetermined path called an orbit. Depending on their orbital distance from the Earth, satellites fall into two main categories, geostationary earth orbit (GEO) and low earth orbit (LEO) [2].

11.2.5.1 GEO satellites

GEO satellites have a circular equatorial (east-west) orbit 36,000 Km above the equator. They rotate at the same speed the Earth rotates around its axis (i.e., they take approximately 24 hours to do one circular orbit). The important implication of this rotation speed is that from a ground station it appears to stay fixed in the sky. Therefore, fixed antennas can be installed at the ground station pointing to a fixed spot in the sky; there is no need for expensive, steerable antennas to continuously track GEO satellites.

Due to their high altitude, each GEO satellite can cover a vast area of the Earth, and global coverage can be achieved with only a handful of GEO

satellites. To be exact, as little as three GEO satellites can provide coverage of the entire Earth.

One of the advantages of using GEO satellites for communication is the uninterrupted connection (line of sight) between the ground station and the satellite 24 hours a day, seven days a week. As we shall see later in this section, this is not possible with LEO satellites. The other advantage of GEO satellites is their large Earth surface coverage; ground stations under the coverage of the same GEO satellite can communicate without being relayed to other satellites in the sky. The stationary nature of GEO satellites with respect to the ground station and the possibility of routes with a minimal number of satellite hops yield a stable delay for GEO satellite-based ATM connections.

The main disadvantage of GEO satellites is a long propagation delay due to their very high altitude. The typical ground-satellite-ground signal propagation delay is about 270 ms. Therefore, the RTT for TCP acknowledgment amounts to a hefty 540 ms over a GEO satellite link.

Another limitation of GEO satellites is that they operate with very high frequencies to cover long distances between the ground station and the satellite. Large antennas are necessary for such high frequencies, making it impractical for mobile users to communicate directly with the satellite using their hand-held devices. Instead, ground stations are equipped with large antennas, and mobile users communicate to the satellite via these ground stations. It is, therefore, difficult to provide 100% coverage of the Earth's surface, as this would mean installing hundreds of thousands of ground stations all over the Earth, on land and sea.

11.2.5.2 LEO satellites

In the early 1990s, under the project known as Iridium [3], Motorola launched a series of LEO satellites orbiting only 750 Km above the Earth's surface to overcome the propagation delay and the high frequency (and large antenna) problem faced by traditional GEO satellites. The typical ground-satellite-ground signal propagation delay for a single hop LEO connection is only about 5 ms. The basic idea of the Iridium project was to provide global telecommunication services using hand-held devices communicating directly with satellites.

Unlike the equatorial (east-west) GEO orbits, LEO satellite orbits are polar orbits (north-south). Unfortunately, although LEOs significantly

decreased the propagation delay, their low altitude led to increased delay variance or jitter due to the following reasons:

- *Handovers:* At low altitude, LEO satellites are not geostationary. To a ground station or user terminal, a LEO satellite looks like a sun, except that the satellite rises and sets in a north-south, rather than an east-west, orientation. To provide 24-hour coverage, multiple satellites are launched in every orbit, so that when one satellite sets in one horizon, the other rises in the opposite horizon. However, this continuity requires a connection handover from the setting satellite to the rising satellite. Given satellite dynamics, such handovers can occur every eight to eleven minutes [4]. During handover, increased delay can be experienced. This increase in delay during handovers causes increased delay variance or jitter.
- *Rerouting:* LEO satellites not only move with respect to Earth stations; they also change locations relative to other satellites in other orbits. Therefore, when multiple satellites from multiple orbits are involved in a given connection, it may be necessary to perform dynamic rerouting of calls to maintain route optimality. Rerouting results in further changes in delays.
- *Packet reordering:* Dynamic rerouting within a call may cause packets to appear out of order at the destination. These out-of-order packets need to be buffered at the edge of the network until the ordering is complete leading to further delay variance.

11.2.6 Some properties of satellite ATM networks

Satellite ATM networks exhibit some properties not commonly found in terrestrial networks. This section discusses these properties.

11.2.6.1 High bandwidth delay product

The product of the link bandwidth and the two-way delay between the sender and the receiver, also known as RTT, represents the amount of data in transit for full link utilization. This product is often called bandwidth delay product (BDP). Large BDP means that the sender needs to have a large window size to buffer a large amount of data in transit.

To appreciate the difference in BDP between terrestrial and satellite ATM networks, let's consider an OC-3c link (155.52-Mbps line

rate) between Melbourne and Perth (between the east and west coasts of Australia). Excluding protocol overhead, the application level throughput for OC-3c is roughly 134 Mbps. For a 3,000-Km terrestrial fiber link, the round-trip propagation delay between Melbourne and Perth is about 30 ms (light travels at a speed of 200 Km/ms in fiber), which gives a BDP of 134 Mbps × 30 ms or roughly 500 KB. Now, compare this with a satellite connection with a typical RTT = 540 ms, which yields a BDP of 134 Mbps × 540 ms, or 9 MB. The buffering requirement for a satellite connection would thus be 18 times that of a terrestrial connection for establishing broadband communications between the east and west coasts of Australia.

11.2.6.2 High BER

A bit error in the communication link will corrupt a data packet and eventually cause the sending TCP to retransmit the packet. Such retransmissions waste bandwidth and force TCP into slow-start, which decreases TCP throughput. A high BER, therefore, is a performance challenge for TCP-based applications.

Fiber-based terrestrial links exhibit a very low BER. The BER for fiber is about 1×10^{-12} or one bit error in every one trillion bits on average. This is a very low BER. Fiber-based terrestrial networks, therefore, do not pose any threat to TCP performance as far as BER is concerned. The primary cause of TCP retransmission in these terrestrial networks is buffer overflow and packet loss during congestion.

Satellite networks, however, have a very high BER due to susceptibility to atmospheric and environmental noise (such as rain and interference with other signals). It is not uncommon to observe a BER as high as 1×10^{-5} in some very noisy satellite channels.

11.2.6.3 Burst error

A more serious problem than the BER is how the bit errors occur. While fiber-based terrestrial links exhibit random errors, satellite links are known for bursty errors where two or more consecutive bits are affected by noise. This is due to variation in satellite link attenuation and the use of convolutional coding to compensate for channel noise [1].

To see why bursty errors are a problem for ATM networks, we need to examine how the ATM header error check (HEC) field works. HEC can detect only single bit errors. An ATM cell header affected by a burst error,

therefore, will go undetected and will eventually be discarded. As a result, burst errors increase the ATM cell discard rate, which, in turn, causes TCP retransmission and loss of goodput.

11.2.7 Standards bodies

A number of organizations are responsible for developing standards for ATM over satellite, identifying TCP enhancements to better utilize TCP connections over satellite links, and providing pointers to research issues in the area of TCP over satellite links. Sections 11.2.7.1–11.2.7.3 discuss these organizations and their progress in standardization activities [5].

11.2.7.1 ITU-R

The ITU-R Working Party 4B is developing two recommendations in alignment with their previous recommendations, Rec I.356: "B-ISDN ATM Layer Cell Transfer Performance" and Rec I.357: "B-ISDN Semi Permanent Connection Availability." The new draft recommendations—"Performance of B-ISDN ATM via Satellite" and "Availability for ATM via Satellite"—consist of reference models for satellite paths, ATM performance objectives for satellites, and translation between the traffic management layer and physical layer. In addition, they describe simulated and measured performance of ATM over satellite and other aspects of ATM over satellite links. The working group is also studying ATM performance objectives for satellite systems, the impact of satellite characteristics on ATM performance, the QoS requirements of ATM services and applications, and ATM availability due to traffic congestion or queuing from on-board ATM equipment, among other topics.

11.2.7.2 TIA

TIA is working on developing a common air interface standard for the GEO mobile satellite service for such issues as interworking with terrestrial GSM, the necessary architecture and guidelines for satellite ATM networks to support different data rates, mobility and on-board processing and switching requirements, and traffic management in ATM networks over satellite links.

11.2.7.3 IETF

The TCP Over Satellite Working Group has produced a "best current practice" RFC [6] that describes the issues affecting TCP throughput over

satellite links. It identifies the domains in which each issue applies, including network topology, satellite orbit (LEO, MEO, and GEO), and link rates; fixes for both protocols and implementations to increase TCP throughput; and areas for further research. The RFC includes consideration of the following issues:

- Transport layer issues affecting TCP over satellite links;
- Existing TCP options;
- Compliant implementations that have some known improved performance over satellite links;
- Recommendations of well understood protocol changes;
- Identification of protocol changes that are potentially promising but require further investigation to be well understood.

The working group has also produced an Internet draft [7] that describes possible TCP enhancements to allow TCP to better utilize the available bandwidth provided by networks containing satellite links. The algorithms and mechanisms outlined have not been judged mature enough to be recommended by the IETF. The goal of the draft is to educate researchers about the current work and progress being done in TCP research related to satellite networks.

11.3 TCP enhancements

Chapter 2 explains how TCP uses slow-start and congestion-avoidance algorithms to adjust its transmission rate in the presence of network congestion. These algorithms, however, may not work well for satellite links for several reasons. For example, the absence of acknowledgment in a satellite environment could mean data error and not necessarily network congestion as interpreted by TCP. In such a case, reducing the TCP transmission rate results in unnecessary loss of throughput. This section discusses several enhancements of the original TCP that can improve TCP performance over satellite links.

11.3.1 Fast retransmit and fast recovery

TCP sends an acknowledgment when a data segment is received. It always acknowledges the highest in-sequence data segment received. If a data segment is not received at the receiver, and a later data segment arrives, the receiver keeps on sending duplicate acknowledgments for the same highest in-sequence data received. Instead of waiting for the RTO timer to expire before retransmitting the lost segment, the fast retransmit algorithm [8] uses three duplicate acknowledgments to detect a lost segment and retransmits the lost segment. To adjust the sending rate, TCP then uses fast recovery [8], which consists of the following steps:

- Reducing *ssthresh* to *cwnd*/2;
- Setting *cwnd* to half of the last value plus the number of duplicate acknowledgments received;
- Entering the congestion avoidance phase to send new data.

Fast retransmit is based on the fact that multiple duplicate acknowledgments indicate nonsevere network congestion and that the TCP transmission rate could thus be reduced to approximately half instead of starting from the beginning as is necessary with slow-start. As a result, the fast retransmit and fast recovery algorithms increase the TCP throughput (over slow-start) by dropping the transmission rate to half the value before loss was detected, entering the congestion avoidance phase, and reducing the waiting time by retransmitting before the expiry of the RTO timer.

Fast retransmit can handle the loss of only one segment per window of data. In the case of multiple losses per window, TCP has to wait for the RTO timer to expire to enter the slow-start phase.

11.3.2 Selective acknowledgment (SACK)

Fast retransmit can increase the TCP throughput in the case of loss of only one segment per window. For multiple segment losses per window of data, TCP enters slow-start resulting in low throughput. Selective acknowledgment (SACK) [9] can be used by the receivers to inform the sender of the exact segment that has to be retransmitted. SACK helps in increasing the TCP throughput over satellite links by avoiding the slow-start phase in the case of multiple segment losses per window of data.

11.3.3 Large sliding window

The maximum throughput of a TCP connection is given by:

$$\text{throughput} = \text{window size}/\text{RTT}$$

With a maximum TCP window size of 65,535 and large RTT delays for satellite links (such as 540 ms for a GEO satellite), the maximum TCP throughput is about 121 Kbytes/sec, which is too small to utilize the bandwidth available in satellite links. Since the RTT cannot be decreased with GEO satellites, the maximum throughput can be increased by increasing the window size. TCP has, therefore, been extended to allow for high performance using large window sizes by using a scaling option.

The window scaling option of TCP [10] extends the 16-bit window field of the TCP header by adding a new three-byte window scaling option field in the TCP header. The scaling option is negotiated during TCP connection setup. It provides a maximum window size of 2^{30} bytes (see Chapter 2), which helps to increase the throughput of TCP connections over long-delay satellite links.

11.3.4 TCP timestamp

TCP uses RTT estimates to determine the value of the RTO timer that is required to detect lost segments. Many TCP implementations base their RTT calculation upon a sample of only one packet per window. For large window sizes, such as those used in conjunction with satellite links, this may result in an undersampling of the RTT measurement leading to inaccurate results for RTT values.

The above problem can be solved by using the TCP timestamp option [10], whereby the sender places timestamps in each data segment. The timestamps are echoed by the receiver to the sender in acknowledgment packets, thereby allowing the sender to obtain a much more accurate estimate of the RTT. Using the TCP timestamp option to obtain better estimates of the RTT is usually referred to in the literature as the RTTM mechanism.

11.3.5 Protection against wrapped sequences (PAWS)

The sequence number field of a TCP connection is finite and wraps around when the maximum limit is reached. There is, therefore, a possibility that an old duplicate segment arrives after the sequence number field has

reached its maximum and wrapped around resulting in the TCP connection getting confused. The protection against wrapped sequences (PAWS) [10] mechanism solves the confusion by using the above mentioned TCP timestamp option. The basic idea is that a segment can be discarded as an old duplicate if it is received with a timestamp value that is less than some timestamp recently received on the connection.

11.3.6 TCP Reno

Various modifications to the original TCP have been made to increase the throughput in the case of long-delay satellite links. The modifications have been incorporated in various versions of TCP. The first version, called TCP Reno, incorporates slow-start, congestion avoidance, fast retransmit, fast recovery, large sliding window, and delayed acknowledgment.

11.3.7 TCP New Reno

The inability of fast retransmit to solve the problem of multiple segment losses in a single window of data is taken care of in TCP New Reno by extending the fast retransmit phase until acknowledgments are received for all segments transmitted until the instant when TCP entered fast retransmit. The *cwnd* is increased by one for every duplicate acknowledgment received. If the acknowledgment received after retransmitting the first lost segment is less than the highest segment transmitted, it retransmits the second lost segment without waiting for three duplicate acknowledgments, and so on for subsequent acknowledgments until the last transmitted segment is acknowledged.

11.4 TCP Performance

11.4.1 Simulation

To evaluate the performance of TCP over satellite ATM networks, Goyal et al [4] performed a detailed simulation study with various TCP enhancements and satellite delays. This section discusses the simulation model, the TCP performance metric, and the simulation results.

11.4.1.1 Simulation model

The satellite network model used for simulation is shown in Figure 11.3. Earth stations (ATM switches) connect TCP/IP hosts via a satellite

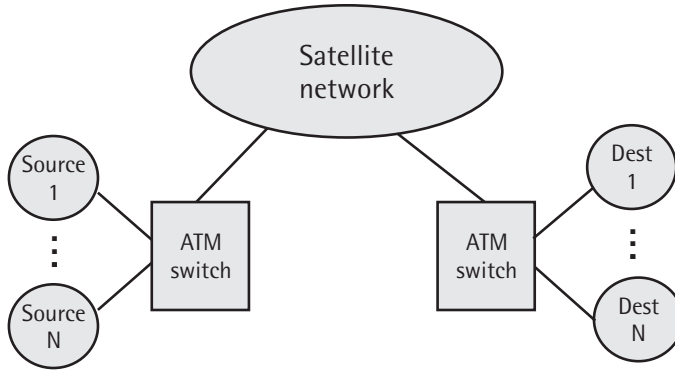


Figure 11.3 Simulation model for TCP over satellite ATM networks.

network. The station that connects the sources is the single bottleneck switch. The goal of this simulation is to determine the buffering requirement at the bottleneck switch to achieve good TCP performance.

All sources are identical and persistent TCP sources. The ACK delay timer is deactivated at the receiver, and the receiver sends an ACK as soon as it receives a segment. The simulation implements a SACK TCP and window scaling option to attain the maximum throughput. All link bandwidths are 155.52 Mbps. Simulations were conducted for three types of latencies, single-hop LEOs, multiple-hop LEOs, and GEOs. Table 11.2 shows the simulation parameters.

The performance metric measured in the simulations is the throughput efficiency of the TCP sources. Efficiency is a normalized measure for the aggregate TCP throughput normalized to the maximum throughput that can be achieved on a 155.52-Mbps link after deducting the bandwidth consumed by the TCP, IP, LLC, AAL5, and ATM headers or trailers. The value for efficiency, therefore, varies between 0 and 1.

11.4.1.2 Simulation results

Simulation results for TCP efficiency values for three different latencies are summarized as follows [4]:

- For very small buffer sizes (less than 0.5 × BDP), TCP efficiency is very low; TCP performance deteriorates as the number of sources increases.

Table 11.2
Simulation Parameters for TCP Over ATM Satellite Networks

Parameter	Value
Number of sources	5, 15, 50, 100
TCP RTT	30 ms (single LEO), 120 ms (multiple LEO), 570 ms (GEO)
Buffer size at bottleneck switch (single LEO)	375, 750, 1,500, 3,000, 6,000, 12,000, 24,000, 36,000 cells
Buffer size at bottleneck switch (multiple LEO)	780, 1,560, 3,125, 6,250, 12,500, 25,000, 5,000, 100,000 cells
Buffer size at bottleneck switch (single GEO)	3,125, 6,250, 25,000, 50,000, 100,000, 200,000, 400,000 cells
Packet discard policy	Selective drop
TCP maximum segment size	9,180 bytes
TCP timer granularity	100 ms

- For buffers greater than 0.5 * BDP, TCP reaches a 98% efficiency, irrespective of the number of sources.

The above results suggest that if the buffering requirement is expressed in BDP, the amount of buffer required to achieve high TCP throughput is only half of the BDP of the end-to-end link, and this result scales well with the number of sources.

In a separate study [11], it was shown that the fairness among the individual TCP connections was also very high due to the selective drop policy implemented in the simulations.

11.4.2 Experiments with NASA's broadband satellite system

NASA has been working for many years with the satellite communications industry to develop advanced commercial satellite communication systems [12]. The explosive growth of the Internet and the enormous business opportunity for commercial communication service providers have recently led NASA to experiment with the commercial protocols, especially TCP/IP and ATM, to better understand the performance dynamics of these protocols over commercial broadband satellite networks. This

section presents the TCP performance results obtained from some of these experiments.

11.4.2.1 Experimental setup

Charalambos [13] reports two experiments involving TCP over ATM using NASA's Advanced Communications Technology Satellite (ACTS) system. The first experiment was conducted in a congestion-free environment (with no packet loss and hence no TCP retransmit) as illustrated in Figure 11.4. The sending and receiving TCP hosts are both connected to the same local ATM switch at the University of Kansas, but the ATM VC that connects these two hosts goes through NASA's ACTS systems. The average RTT on this VC is measured at 534 ms using standard PING packets. There is no congestion in the end-to-end path, as all links are at least OC-3c links operating at 155 Mbps.

To observe TCP/ATM performance over satellites under network congestion, a second experiment was conducted with three sending hosts with persistent TCP sources connected to the local switch all using OC-3c links (see Figure 11.5). Under these conditions, the ATM network becomes congested. Packets are dropped when the switch buffer is overflowed. Consequently, TCP retransmissions are observed.

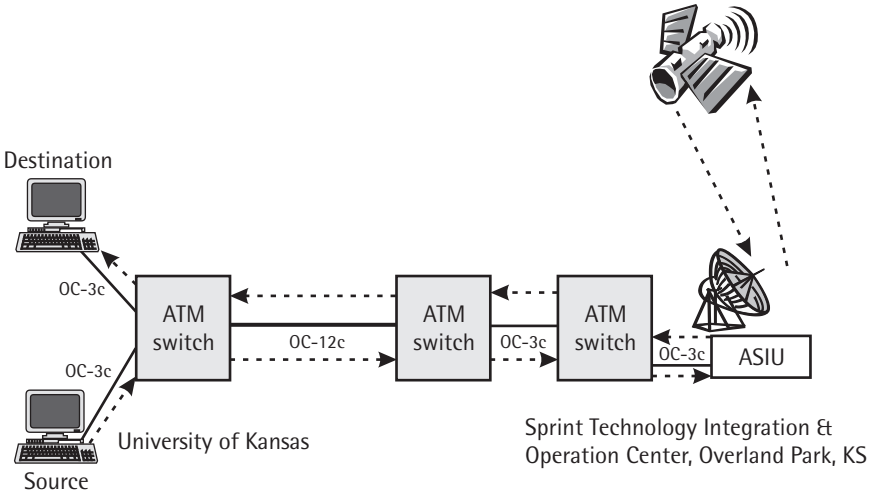


Figure 11.4 Congestion-free configuration for TCP over satellite ATM experiments.

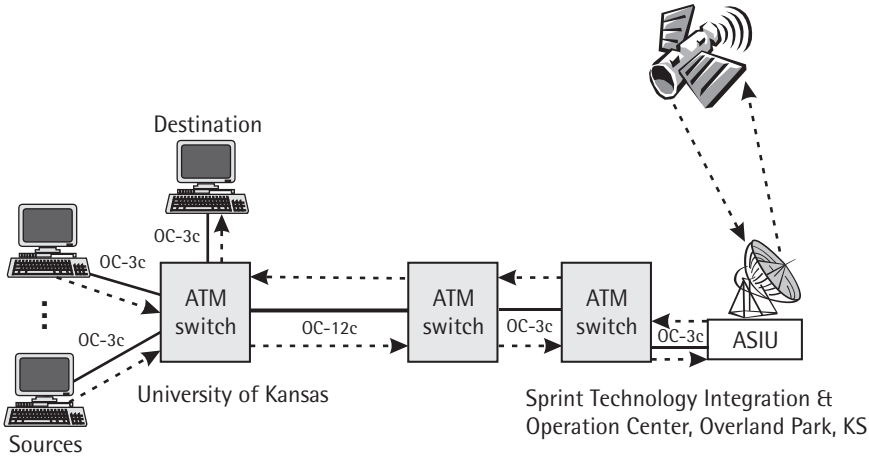


Figure 11.5 Congested network configuration for TCP over satellite ATM experiments.

All three TCP varieties, TCP Reno, TCP New Reno, and TCP SACK, were tested in each of the above experiments. The MTU used for IP over ATM was 9,180 bytes. Section 11.4.2.2 presents TCP throughput results obtained from the above experiments.

11.4.2.2 TCP performance results

Congestion-free experiment Table 11.3 shows the maximum and average TCP throughput from 10 replications. The channel utilization is the average TCP throughput normalized to the theoretical maximum TCP throughput. Note that the maximum theoretical TCP throughput possible over a 155.52-Mbps line rate is slightly less than 135.12 Mbps—the maximum throughput at the IP level for 9,180-byte IP datagrams is 135.12 Mbps; see Table 4.2—it is 134.513 Mbps to be exact.

The experimental results show that even in a congestion-free network, TCP is unable to achieve 100% link utilization. The other important result is that TCP SACK performs better than both TCP Reno and TCP New Reno. Charalambos [13] explains that 100% utilization is not reached due to the use of different operating systems in the source and destination hosts. It was not confirmed whether 100% channel utilization could be achieved with any TCP flavor over the satellite connection.

Table 11.3
TCP Performance Over Congestion-Free Satellite ATM Networks

Throughput (Mbps)	TCP Reno	TCP New Reno	TCP SACK
Maximum	119.00	123.27	126.76
Average	109.72	121.24	121.27
Channel utilization	0.82	0.90	0.90

Congested network experiment Table 11.4 shows TCP performance results for the congested scenario. As expected, TCP performance was much poorer than in the congestion-free case. TCP SACK again performed the best but was able to achieve only 60% utilization.

11.4.2.3 TCP performance over noisy satellite links

The BER observed for the above experiments as 1.0×10^{-12}, which is comparable to terrestrial fiber links. For many satellite links, however, the BER is much higher. For such noisy satellite links, TCP performance will still suffer even if there is no congestion, as high BERs will cause packet corruption and TCP retransmission.

Charalambos [13] evaluated TCP/ATM performance over noisy satellite links by artificially introducing different BERs in the link. TCP performance, as a percentage of the optimal throughput (which is 134.513 Mbps in the experimental network), for different BERs is shown in Table 11.5 for three TCP flavors. It is interesting to note that none of the TCP extensions can cope with high BER in the satellite link. Even with TCP SACK, channel utilization is only about 1% of the optimal TCP throughput for a BER of 8.9×10^{-4}.

Table 11.4
TCP Performance Over Congested Satellite ATM Networks

	TCP Reno	TCP New Reno	TCP SACK
Average throughput (Mbps)	54.11	65.62	81.15
Channel utilization	0.40	0.49	0.60

Table 11.5
TCP/ATM Performance as a Percentage of Optimal Throughput for Noisy Satellite Links

BER	TCP Reno	TCP New Reno	TCP SACK
8.9×10^{-4}	1.06%	1.13%	1.45%
0.8×10^{-7}	1.14%	1.24%	1.57%
1.3×10^{-8}	3.81%	4.84%	5.58%
1.1×10^{-9}	12.15%	17.18%	19.75%

11.5 Summary

Alongside the terrestrial networks, satellite ATM networks will play a vital role in building the backbones of the TCP/IP networks on a global scale. Two major features distinguishing satellite networks from terrestrial networks in terms of their impact on TCP performance are a very large RTT and a high BER. Several TCP enhancements have recently been standardized, or are in the process of standardization, to address TCP performance problems for such large RTT and high BER links. Experimental results show that among the various TCP enhancements, TCP SACK is the most promising for satellite communications when the BER is not too high. None of the TCP enhancements, however, appears to be robust against a high BER. TCP throughput slumps to a mere 1% of the optimal value when the BER exceeds 1×10^{-7}.

TCP performance over congested satellite ATM networks is very poor. Large switch buffers are necessary for large RTT links to achieve good performance. Simulation studies show that TCP can achieve 98% of channel utilization for a buffer capacity able to hold data transmitted in $0.5 \times$ RTT.

References

[1] Akyildiz, I. F., and J. Seong-Ho, "Satellite ATM Networks: A Survey," *IEEE Communications Magazine,* Vol. 35, No. 7, July 1997, pp. 30–43.

[2] Tanenbaum, A. S., *Computer Networks, Third Edition,* Englewood Cliffs, NJ: Prentice Hall, 1996.

[3] Iridium project, http://www.iridium.com/.

[4] Goyal, R., S. Kota, R. Jain, S. Fahmy, B. Vandalore, and J. Kallaus, "Analysis and Simulation of Delay and Buffer Requirements of Satellite ATM Networks for TCP/IP Traffic," submitted to *IEEE Journal of Selected Areas in Communications,* March 1998, available from http://www.cis.ohio-state.edu/~jain/papers/jsac98.htm.

[5] Kota, S., T. vonDeak, R. Jain, and F. Yegenoglu, "A Progress Report on the Standards Development for Satellite ATM Networks," ATM Forum Contribution 98-0828, November 1998.

[6] Allman, M., D. Glover, and L. Sanchez., "Enhancing TCP Over Satellite Channels Using Standard Mechanisms," *Internet RFC 2488,* IETF, January 1999.

[7] Allman, M., D. Glover, J. Griner, et al, "Ongoing TCP Research Related to satellites," Internet draft, draft-ietf-tcpsat-res-issues-12.txt, October 1999.

[8] Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *Internet RFC 2001*, January 1997.

[9] Mathis, M., J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options," *Internet RFC 2018,* IETF, October 1996.

[10] Jacobson, V., R. Braden, and D. Borman, "TCP Extensions for High Performance," *Internet RFC 1323,* IETF, May 1992.

[11] Kota, S., R. Goyal, and R. Jain, "Satellite ATM Network Architectural Considerations and TCP/IP Performance," Proceedings of the 3rd K-A Band Utilization Conference, 1997.

[12] Ivancic, W. D., D. Brooks, B. Frantz, D. Hoder, D. Shell, and D. Beering, "NASA's Broadband Satellite Networking Research," *IEEE Communications Magazine,* Vol. 37, No. 7, July 1999, pages 40–47.

[13] Charalambos, C. P., "Performance Evaluation and Analysis of TCP on ATM Over High Bandwidth Delay Product Networks," Masters Thesis, Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS.

About the Authors

Dr. Mahbub Hassan (mahbub@ieee.org) is a senior lecturer in the School of Computer Science and Software Engineering at Monash University in Melbourne, Australia. He received his Ph.D. in computer science from Monash University and his M.Sc. in computer science from the University of Victoria, Canada. He serves on the editorial board of *IEEE Communications Magazine*. His research interests include internetworking architectures and protocols for TCP/IP and ATM networks. He has organized conference sessions and has guest edited special journal issues on TCP/IP network performance. In addition to conducting teaching and research at Monash University, Dr. Hassan also serves as a consultant to industry and government. Further information on Dr. Hassan's research and professional activities can be found on the web at http://www.sd.monash.edu.au/~mahbub/.

Dr. Mohammed Atiquzzaman (atiq@ieee.org) received his M.Sc. and Ph.D. degrees in electrical engineering from the University of Manchester Institute of Science and Technology, England. He is currently a faculty member in the Department of Electrical & Computer Engineering at the

University of Dayton, Ohio. He has previously held faculty positions in Australia.

Dr. Atiquzzaman is a senior editor of *IEEE Communications Magazine* and a member of the editorial boards for *Computer Communication Journal*, *Telecommunications Systems Journal*, and *Real-Time Imaging Journal*. He has guest edited many special issues of various journals with feature topics like Switching and Traffic Management for Multimedia (*IEEE Communications Magazine*); Architecture, Protocol, and Quality of Service (*European Transactions on Telecommunications*); ATM Switching and ATM Networks (*International Journal of Computer Systems Science & Engineering*); and Parallel Computing on Clusters of Workstations (*Parallel Computing*). He has also organized conference sessions and has served on the technical program committees of many national and international conferences including *IEEE INFOCOM*, *IEEE Globecom*, and the *IEEE Annual Conference on Local Computer Networks*. He is a senior member of IEEE.

Dr. Atiquzzaman's current research interests are in next generation Internet, broadband networks, multimedia over high-speed networks, and TCP/IP over ATM. He has taught many short courses for industry professionals on computer and telecommunications networking. He has over 100 refereed publications in the above areas, most of which can be accessed at http://www.engr.udayton.edu/faculty/matiquzz/.

Index