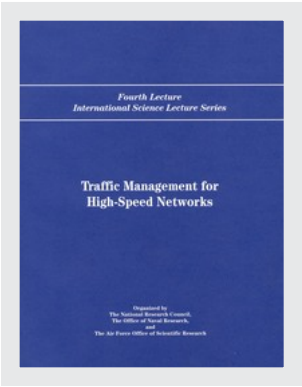

Traffic Management for High-Speed Networks: Fourth Lecture International Science Lecture Series (1997)

DETAILS

32 pages | 8.5 x 11 | PAPERBACK
ISBN 978-0-309-05798-1 | DOI 10.17226/5769

CONTRIBUTORS

by H.T. Kung, Gordon McKay Professor of Electrical Engineering and Computer Science, Harvard University

SUGGESTED CITATION

National Research Council 1997. *Traffic Management for High-Speed Networks: Fourth Lecture International Science Lecture Series*. Washington, DC: The National Academies Press. https://doi.org/10.17226/5769.

Traffic Management For High-Speed Networks

by
H.T. Kung
Gordon McKay Professor of Electrical Engineering and Computer Science
Harvard University

National Academy Press
Washington, D.C. 1997

The National Research Council serves as an independent advisor to the federal government on scientific and technical questions of national importance. Established in 1916 under the congressional charter of the private, nonprofit National Academy of Sciences, the Research Council brings the resources of the entire scientific and technical community to bear on national problems through its volunteer advisory committees. Today the Research Council stands as the principal operating agency of both the National Academy of Sciences and the National Academy of Engineering and is administered jointly by the two academies and the Institute of Medicine. The National Academy of Engineering and the Institute of Medicine were established in 1964 and 1970, respectively, under the charter of the National Academy of Sciences.

The National Research Council has numerous operating units. One of these is the Naval Studies Board, which is charged with conducting and reporting upon surveys and studies in the field of scientific research and development applicable to the operation and function of the Navy.

A portion of the work done to prepare this document was performed under Department of Navy Grant N00014-94-1-0200 issued by the Office of Naval Research and the Air Force Office of Scientific Research under contract authority NR 201-124. However, the content does not necessarily reflect the position or the policy of the Department of the Navy or the government, and no official endorsement should be inferred.

Additional copies of this report available from:
Naval Studies Board
National Research Council
2101 Constitution Avenue, N.W.
Washington, D.C. 20418
Printed in the United States of America

NAVAL STUDIES BOARD

Navy Liaison Representatives

Staff

COMMISSION ON PHYSICAL SCIENCES, MATHEMATICS, AND APPLICATIONS

Preface

The International Science Lecture Series (ISLS) operates as a special project of the National Research Council's Commission on Physical Sciences, Mathematics, and Applications. The series was established in 1990 at the request of the Office of Naval Research (ONR) and joined in 1992 by the Air Force Office of Scientific Research (AFOSR). The purpose of the series is to advance communication and cooperation within the international scientific community. A search committee established by the National Research Council (NRC) selects prominent U.S. scientists to lecture in three areas of basic scientific inquiry: ocean and meteorological sciences, materials science, and information science. The countries in which the lectures are to be given are selected on the basis of consultations with the international scientific community, with the science attache in U.S. embassies, with senior representatives of ONR-Asia and ONR-Europe, and with both ONR and AFOSR representatives in Washington, D.C. Wherever appropriate, each lecture in a host country is followed by formal and informal discussions with senior government, industrial, and academic representatives to expand the dialogue on research progress, problems, and areas of common interest in order to identify research opportunities that lend themselves to greater cooperation and collaborative effort. Following each tour, the formal lecture is published for wider international distribution.

The fourth lecture of the series, which is presented here, is *Traffic Management for High-Speed Networks* by H.T. Kung, Gordon McKay Professor of Electrical Engineering and Computer Science, Harvard University. The first lecture in the series, *The Heard Island Experiment*, was presented by Walter H. Munk, holder of the Secretary of the Navy Research Chair at the Scripps Institution of Oceanography, University of California at San Diego, and the second lecture, *Fountainhead for New Technologies and New Science*, was presented by Rustum Roy, Evan Pugh Professor of the Solid State and professor of geochemistry, Pennsylvania State University. The third lecturer was John E. Hopcroft, who is the Joseph C. Ford Professor of Computer Science at Cornell University and who gave the lecture, *Computing, Communication, and the Information Age*.

Professor Kung's lecture tour consisted of two separate trips—one in the Far East and the other in Siberia. He gave his lecture first at the Chinese University of Hong Kong on June 5, 1995, to the computer sciences community. While in Hong Kong, Professor Kung and the ISLS representatives from the NRC, ONR, and AFOSR also visited the Hong Kong University of Science and Technology and the Hong Kong University. Professor Kung delivered his lecture at these two institutions as well. On June 8 he presented his lecture at the Sino-American Joint Seminar on Trends in Information Science held in Beijing, China. Discussions also were held with the staffs and faculties of Tsing Hua University and Peking University. Professor Kung visited Fudan University and Shanghai Jiaotong University on June 14 and 15 and Zhejiang University in Hanzhou on June 16. He gave his lecture at Fudan University.

The second tour took Professor Kung and the ISLS group to Novosibirsk, Siberia, in January 1996. They met on January 8, 1996, with the staff of the A.P. Ershov Institute of Informatics

Systems and discussed future working relationships between the U.S. and Russian information sciences research communities. On January 9, the ISLS group met with the staffs of the Institute of Automation and Electrometry as well as the Institute of Computational Technologies. Professor Kung presented his lecture there on January 10 and then visited the Institute for Information Systems and the Novosibirsk State University.

The National Research Council, the Office of Naval Research, and the Air Force Office of Scientific Research would like to express their appreciation to the many host-country representatives for their hospitality and their invaluable assistance in arranging Professor Kung's visits and the many discussions that followed the formal lecture. The sponsors are also indebted to the American Embassy representatives in each of the host countries and to the representatives of ONR-Asia and ONR-Europe for their tireless efforts to make the lecture tours a success.

Contents

Traffic Management for High-Speed Networks

Abstract

Network congestion will increase as network speed increases. New control methods are needed, especially for handling "bursty" traffic expected in very high speed networks such as asynchronous transfer mode (ATM) networks. Users should have instant access to all available network bandwidth when they need it, while being assured that the chance of losing data in the presence of congestion will be negligible. At the same time, high network utilization must be achieved, and services requiting guaranteed performance must be accommodated. This paper discusses these issues and describes congestion control solutions under study at Harvard University and elsewhere. Motivations, theory, and experimental results are presented.

Why New Control Methods are Needed

RAPID INCREASE IN NETWORK SPEEDS

Over the past decade, the speed of computer and telecommunications networks has improved substantially. To wit:

- 1980s

- 1.5-Mbps (megabits per second) T1
- 4- or 16-Mbps Token Rings
- 10-Mbps Ethernet

- 1990s

- 45-Mbps T3
- 100-Mbps Ethernet
- 100-Mbps FDDI
- 155-Mbps OC-3 ATM
- 622-Mbps OC-12 ATM

This rapid growth in speed is expected to continue over the next decade, because many new applications in important areas such as data and video will demand very high network bandwidths. These high-speed networks are introducing major new challenges in network congestion control, as explained in the next two sections. That the high-speed networks would make the solution for network congestion harder is contrary to what one's intuition might suggest.

NETWORK CONGESTION PROBLEM

Any network has bottlenecks or congestion points, i.e., locations where more data may arrive than the network can carry. A common cause for congestion is a mismatch in speed between networks. For example, a typical high-performance local area network (LAN) environment in the next several years may have the architecture shown in Figure 1. While the servers will use new high-speed asynchronous transfer mode (ATM) connections at the OC-3 rate of 155 Mbps, many clients will still depend on old, inexpensive but slower, 10-Mbps Ethernet connections. Data flowing from the servers at 155 Mbps to the clients at 10 Mbps will experience congestion at the interface between the ATM and Ethernet networks.

Congestion can also occur inside a network node that has multiple ports. Such a node can be a switch such as an ATM switch or a gateway such as a router. As depicted in Figure 2, congestion arises when data, destined for a single output port, arrive at many different input ports. The faster and more numerous these input ports are, the severer the congestion will be.

A consequence of congestion is the loss of data due to buffer overflow. For data communications in which every bit must be transmitted correctly, lost data will have to be retransmitted, and will result in degraded network utilization and increased communications delay for end users.

INADEQUACY OF BRUTE-FORCE APPROACH TO PROVIDING LARGE BUFFERS

A universal solution to the problem of losing data because of congestion involves buffer memory in which a congested point can temporarily queue data directed at overloaded output ports. This use of buffer is illustrated in Figure 2. However, simply providing large buffers would likely incur prohibitively high memory cost for high-speed networks, because as network speed increases, so also will the following factors:

- *Buffer overloading rate*. Suppose that data from multiple input ports feed to a single output port, and that all the ports are of the same speed. If all these ports now increase their speed by a factor of *X*, then the overloading rate to the node buffer will also increase

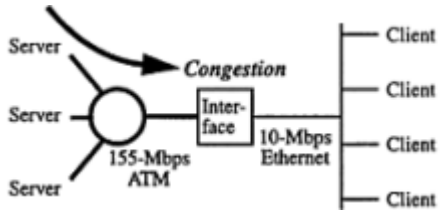

Figure 1 Congestion due to a mismatch in speed between 155-Mbps ATM network and 10-Mbps Ethernet.

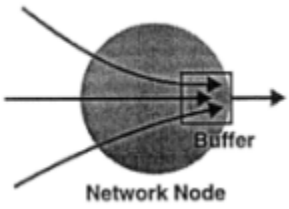

Figure 2 Congestion, in a switch or gateway, due to multiple arrivals at the same output.

by the same factor of X. If, to prevent buffer overflow, the same congestion control scheme is used as was used before, then the feedback delay in the control system, which is a function of propagation delays and largely independent of link speed, will remain essentially the same. An increase in link speed will therefore demand an *X*-fold increase in the required buffer size, if possible data loss is to be kept to the same level as before.

- *Packet or burst size*. For high-speed networks, high-level protocols will use data packets with an increased number of bytes, in order to reduce packet processing overhead at end systems, such as the packet interrupt frequency at receiving hosts. These large packets introduce large bursts of data that may arrive at congestion points at the same time. Assuming the same average load as before, bursts of increased size imply increased overlapping of arriving bursts at congestion points. A larger buffer is thus needed to accommodate these simultaneously arriving large bursts.
- *Transient traffic*. Typical Transmission Control Protocol (TCP) sessions involve a few dozen kilobytes [19], and the required transmission time on an OC-3 link at 155 Mbps is only a few milliseconds. (A survey of Unix file sizes [7] has also shown a similar result for file sizes. That is, the average file length is only around 22 kbytes, and most files are smaller than 2 kbytes.) Thus, for high-speed networks, these sessions will not be long enough to achieve steady-state traffic flow beyond a local or metropolitan area. When facing this type of transient traffic over a wide area, traditional end-to-end flow control methods such as TCP will incur relatively long feedback control delays, and thus such methods cannot be effective in reducing buffer usage inside a network.
- *Bandwidth mismatch*. As new networks are deployed, many of the relatively old, inexpensive, low-bandwidth networks will still be in use. As these new networks with higher and higher speeds emerge, gaps in speed between old and new networks will increase. For handling the same load, this greater mismatch in bandwidth again implies the need for larger buffers.
- *Load speed from computer sources*. A single workstation or personal computer can now consume the whole bandwidth of an OC-3 link. High-end computers such as servers tend to support high-bandwidth network interfaces that run as fast as the fastest computer networks available. One can expect that, at any point in time in the foreseeable future, several high-performance computers, if not just one, will always be able to saturate the fastest links in any network.

To prevent data loss due to congestion, network buffers could be increased to accommodate the increase in each of the above factors. But these factors increase independently, and the multiplicative effects of such increases will demand enormously large buffers. In addition, as network usage increases, so also will the expected number of active sessions on the network and their peak bandwidths. For each session, a network node may have to buffer all the on-the-fly data from a distant sending host to itself when congestion occurs. The buffers occupied by the session can be the entire TCP window if TCP is used. If there are *N* sessions, *N* times the size/capacity of this buffer will be needed.

For all these reasons, brute-force methods of using larger and larger buffers cannot solve the congestion problems to be expected with high-speed networks.

Use of Flow Control

When congestion persists, no amount of buffering is sufficient in the longer term; instead, each source of traffic flowing through a bottleneck must be persuaded to send no more than its fair share of the bottleneck's capacity. That is, proper flow control can bound the buffer requirement.

This is fundamentally a feedback control problem, and many control ideas and principles apply. As depicted in Figure 3, each network node, which can be switches or gateways, collects information about congestion, and informs, directly or indirectly, the sources of data. This feedback is usually based on the amount of buffer space available or in use in the node. The sources act to control how much data they send. This control loop has a delay equal to at least twice the propagation delay between the switch and control point.

Control systems should seek to minimize this delay in feedback, since nodes will need to buffer any data that arrive after the nodes signal the congestion status but before the end of the delay. Moreover, the feedback control delay should be sufficiently small so that the control system can respond in time to any changes in traffic load.

CONTROL OF CONGESTION FOR ATM NETWORKS

Control of congestion for ATM networks is of particular interest, because such networks support very high speed connections and multimedia services. An ATM network can simultaneously support multiple types of services of various qualities. As illustrated in Figure 4, these include constant bit rate (CBR) services for voice and other fixed-rate guaranteed traffic; variable bit rate (VBR) services for video; and available bit rate (ABR) services for data.

Being able to support ABR for data communications represents a major advantage of ATM networks over traditional time division multiplexing (TDM) networks. Under ABR services, users can have instant access to available network bandwidth when they need it, and they do not have to hold onto unused bandwidth when they do not need it. These services are exactly what many computer users desire.

In order to realize this potential of ABR services for data applications, nodes or end systems in a network need to receive status information on buffer or bandwidth usages from downstream entities. That is, effective and efficient flow control is essential.

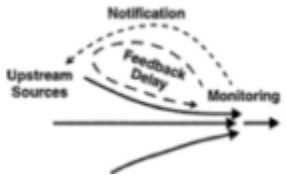

Figure 3 Use of feedback control to handle. Congestion.

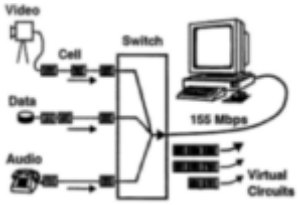

Figure 4 ATM network.

TECHNICAL GOALS OF FLOW CONTROL FOR SUPPORTING ATM ABR SERVICES

Flow control mechanisms designed to support ATM ABR services should meet a variety of technical goals, including the following:

- Data should rarely, if ever, be discarded due to exhaustion of node buffer memory. As mentioned above, such data may have to be retransmitted after a possibly lengthy time-out period, further contributing to network congestion and the delay experienced by the user.
- Network links should be used at full capacity whenever possible. For instance, if one connection sharing a link reduces the rate at which it sends, the others should increase their rates as soon as possible. In particular, as illustrated in Figure 5, the flow control mechanism should allow ABR traffic to fill in, instantly, unused bandwidth left on the link after guaranteed traffic is served.
- All the connections that are constrained by a bottleneck link should get fair shares of that link.
- The flow control mechanism should be robust. Loss or delay of control messages, and admission of additional connections while maintaining the total traffic load, for instance, should not cause increased congestion.
- The network administrator should not have to adjust any complex parameters to achieve high performance.
- The flow control mechanism should have a cost commensurate with the benefits it provides.

Generally speaking, some existing LANs such as Ethernets have satisfied these goals. This explains at least partially why they have been used widely for data applications.

New high-speed networks, such as ATM networks and switched Ethernets, use switches to achieve high performance. They are unlike conventional Ethernets, which use shared media. End systems on switch-based networks cannot monitor network congestion as easily as can end systems on shared-medium networks. Designing flow control schemes to satisfy the above technical goals for these new switch-based networks—especially for wide area networks (WANs)—is a significant challenge.

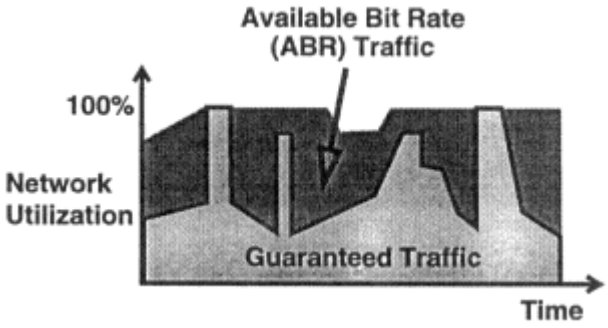

Figure 5 Available bit rate (ABR traffic filling in bandwidth slack left by guaranteed traffic, to maximize network utilization).

TWO TRAFFIC MODELS

Any prediction of how well a flow control scheme will work requires a model for the behavior of network traffic. A full-blown model might involve characteristics of applications and higher-level protocols. For our discussion here, it is enough to distinguish between *smooth* and "*bursty*" traffic.

A smooth traffic source offers a constant and predictable load, or changes only on time scales that are large compared to the amount of time the flow control mechanism takes to respond. Such traffic is easy to handle well; the sources can be assigned rates corresponding to fair shares of the bottleneck bandwidth with little risk that some of them will stop sending and lead to underutilized links. Furthermore, switches can use a small amount of memory, since bursts in traffic intensity are rare.

Sources of smooth traffic include voice and video with fixed-rate compression. The aggregate effect of a large number of bursty sources may also be smooth, particularly in a WAN where the loads from a large number of traffic streams are aggregated and the individual sources have relatively low bandwidth and arc uncorrelated.

Bursty traffic, in contrast, lacks any of the predictability of smooth traffic, as observed in some computer communications traffic [8]. Some kinds of bursts stem from users and applications. A World Wide Web browser clicking on a link, for instance, wants to see a page or image as soon as possible. The network cannot predict when the clicks will occur, nor should it smooth out the resulting traffic, since doing so would hurt the user's interactive response.

Other sources of bursts result from network protocols that break up transfers into individual packets, windows, or RPCs, which are sent at irregular intervals. These bursts are sporadic and typically do not last long enough on a high-speed link to reach steady state over the link round-trip time.

Designing flow control systems for bursty traffic is obviously much more difficult than designing control systems for smooth traffic. In supporting computer communications, which are generally bursty, we will have no choice but to face the challenge of designing effective flow control for bursty traffic.

A FLOOD CONTROL PRINCIPLE

An old principle for controlling floods suggests an approach to controlling network congestion. Dams on a river for holding floods are analogous to buffers in a network for holding excessive data.

To control floods, dams are often built in series along a river, so that an upstream dam can share the load for any downstream dam. As depicted in Figure 6, whenever a dam is becoming full, its upstream dams are notified to hold additional water. In this way, all the upstream dams can help reduce flooding at a downstream congestion point, and each upstream dam can help prevent flooding at all downstream congestion points. The capacity of every dam is efficiently used.

Credit-Based Flow Control

An efficient way of implementing flow-controlled ATM networks is through the use of credit-based, link-by-link, per-VC (virtual circuit) flow control [12, 14, 17]. As depicted in Figure 7, credit-based control works like the method of controlling floods described above. Each

downstream-to-upstream notification is implemented with a credit record message. The scheme generally works over a VC link as follows. A link can be a physical link connecting two adjacent nodes, or a virtual circuit connecting two remote nodes. Before forwarding any data cell over the link, the sender needs to receive credits for the VC from the receiver. At various times, the receiver sends credits to the sender indicating availability of buffer space for receiving data cells of the VC. After having received credits, the sender is eligible to forward some number of data cells of the VC to the receiver according to the received credit information. Each time the sender forwards a data cell of a VC, it decreases its current credit balance for the VC by one.

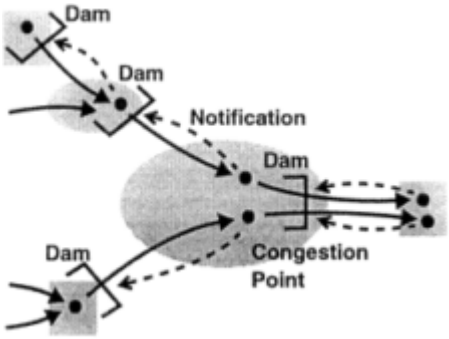

Figure 6 Flood control analogy. All the dams (or buffers) on the path leading to the congestion point can help prevent flooding (or cell loss). Notifications are denoted by dashed arrows.

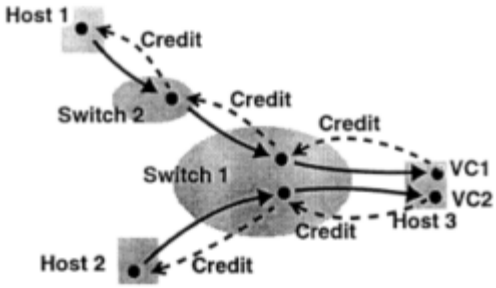

Figure 7 Credit-based flow control. Black dots stand for virtual circuit (VC) buffers.

There are two phases in flow controlling a VC [15]. In the first *buffer allocation* phase, the VC is given an allocation of buffer memory in the receiver. In the second *credit control* phase, the sender maintains a non-negative credit balance to ensure no overflow of the allocated buffer in the receiver.

CREDIT UPDATE PROTOCOL

The Credit Update Protocol (CUP) [12] is an efficient and robust protocol for implementing credit control over a link. As depicted in Figure 8, for each flow-controlled VC the sender keeps a running total Tx_Cnt of all the data cells it has transmitted, and the receiver keeps a running total Fwd_Cnt of all the data cells it has forwarded. (If cells arc allowed to be dropped within the receiver, Fwd_Cnt will also count these dropped cells.) The receiver will enclose the up-to-date value of Fwd_Cnt in each credit record transmitted upstream via a credit cell. When the sender receives the credit record with value Fwd_Cnt, it will update the credit balance, Crd_Bal, for the VC:

$$\text{Crd\_Bal} = \text{Buf\_Alloc} - (\text{Tx\_Cnt} - \text{Fwd\_Cnt}) \qquad (1)$$

where Buf_Alloc is the total number of cells allocated to the VC in the receiver.

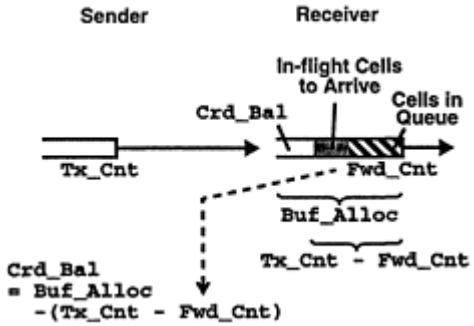

Figure 8 Credit Update Protocol (CUP). The dotted arrow indicates a credit record, transmitted upstream, containing the current value of Fwd_Cnt at the receiver.

Note that the quantity computed by the sender, Tx_Cnt - Fwd_Cnt, represents the "outstanding credits" corresponding to those cells of the VC that the sender has transmuted but the receiver has not founded. As depicted in Figure 8, these cells are "in-flight cells to arrive" and "cells in queue" at the time when the receiver sends credit record Fwd_Cnt to the sender. Thus Crd_Bal computed by the sender using Equation (1) is the proper new credit balance, in the sense that, as long as the sunder transmits no more than Crd_Bal cells, it will not overrun the VC's allocated buffer in the receiver. See [12] for a scheme of using *credit_check cells* periodically sent from the sender to the receiver, to recover from possible loss of data or credit cells due to link errors.

The frequency at which the receiver sends credit records for a VC depends on the VC's progress. More precisely, each time after the receiver has founded a certain number of cells, "N2" cells [14] for some positive integer N2, the receiver will send a credit record upstream. The value of N2 can be set statically or adaptively (see Equation (4) below).

The Buf_Alloc value given to a VC determines the maximum bandwidth allowed to the VC by credit flow control Without loss of generality, we assume that the maximal peak bandwidth of any link is 1, and represent the rate of a VC as a fraction of 1. For the rest of this section, we also make a simplifying assumption that all links have the same peak bandwidth of 1. Let *RTT* be the round-trip time, in cell transmission time, of the link between the sender and the receiver (see Figure 8), including both link propagation delays and credit processing time. Assume that the receiver uses a fair scheduling policy between VCs with Crd_Bal > 0, when forwarding cells out from its output link. Then, if there are *N* active VCs competing for the same output link, the maximum average bandwidth over *RTT* that the VC can achieve is

$$BW = \text{Buf\_Alloc} / (RTT + N2*N) \qquad (2)$$

Note that when there is only one VC using the output port, i.e., $N = 1$, the VC's bandwidth (BW) can be as high as Buf_Alloc / ($RTT + N2$).

The CUP scheme is a lower-level and lighter-weight protocol than are typical sliding window protocols used in, e.g., X.25 and TCP. In particular, CUP is not linked to retransmission of lost packets. In X.25 or TCP, loss of any packet will stop the advancing window until the dropped packet has been retransmitted successfully. To implement this, each data packet carries a

sequence number. In contrast, in CUP the sender does not retransmit lost data cells, the receiver does not reorder received cells, and data cells do not carry sequence numbers. This simplicity is made possible because CUP need only work for ATM VCs that preserve cell order.

It can be shown [14] that CUP produces the same buffer management results as the well-known "incremental" credit updating methods (see, e.g., [7, 9]). In these other methods, instead of sending Fwd_Cnt values upstream the receiver sends incremental credit values to be added to Crd_Bal at the sender.

STATIC VS. ADAPTIVE CREDIT CONTROL

We call a credit-based flow control either *static* or *adaptive* depending on whether the buffer allocation is static or adaptive. In a static credit control, a fixed value of Buf_Alloc is used for the lifetime of a VC. Requiring only the implementation of CUP, or some equivalent protocol, the method is extremely simple.

There are situations, however, where adaptive credit control is desirable. In order to allow a VC to operate at a high rate, Equation (2) implies that Buf_Alloc must be large relative to $RTT + N2*N$. Allocating a small buffer to a VC can prevent the VC from using otherwise available link bandwidth. On the other hand, committing a large buffer to a VC can be wasteful, because sometimes the VC may not have sufficient data, or may not be able to get enough scheduling slots, to transmit at the desired high rate. The proper rate at which a VC can transmit depends on the behavior of traffic sources, competing traffic, scheduling policy, and other factors, all of which can change dynamically or may not be known a priori. In this case, adaptive credit control, which is static credit control plus adaptive adjustment of Buf_Alloc of a VC according to its current bandwidth usage, can be attractive.

Generally speaking, for configurations where a large Buf_Alloc relative to $RTT + N2*N$ is not prohibitively expensive, it may be simplest just to implement static credit control. This would give excellent performance. Otherwise, some adaptive buffer allocation scheme, as described below, may be used to adjust Buf_Alloc adaptively. To maximize flexibility, the adaptation can be carried out by software.

ADAPTIVE BUFFER ALLOCATION

Adaptive buffer allocation allows multiple VCs to share the same buffer pool in the receiver node adaptively, according to their needs. That is, Buf_Alloc of a VC is automatically decreased if the VC does not have sufficient data to forward, cannot get sufficient scheduling slots, or is back-pressured due to downstream congestion. The freed-up buffer space is automatically assigned to other VCs that have data to forward and are not congested downstream.

Adaptive buffer allocation can be implemented at the sender or receiver node. As depicted in Figure 9, in a sender-oriented adaptive scheme [12, 17] the sender adaptively allocates a shared input-buffer at the receiver among a number of VCs from the sender that share the same buffer pool. The sender can allocate buffer for the VCs based on their measured, relative bandwidth usage on the output port p [12].

Receiver-oriented adaptation [13] is depicted by Figure 10. The receiver adaptively allocates a shared output-buffer among a number of VCs from one or more senders that share the same buffer pool. The receiver can allocate buffer for the VCs based on their measured, relative bandwidth usage on the output port q [13].

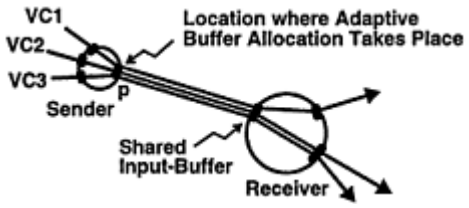

Figure 9 Sender-oriented adaptation. The circles are switches. Each darkened bar denotes a switch point.

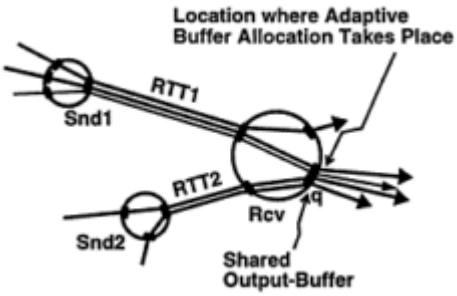

Figure 10 Receiver-oriented adaptation.

Receiver-oriented adaptation is suited for the case where a common buffer pool in a receiver is shared by VCs from *multiple* upstream nodes. Figure 10 depicts such a scenario: the buffer pool at output port q of the receiver switch Rcv is shared by four VCs from two switches Snd1 and Snd2. Note that the receiver (Rcv for Figure 10) can observe the bandwidth usage of the VCs from all the senders (that is, Snd1 and Snd2). In contrast, each sender can observe the bandwidth usage only of those VCs going out from the same sender. Therefore, it is natural to use receiver-oriented adaptation in this case.

Moreover, receiver-oriented adaptation naturally supports the adaptation of $N2$ values for individual VCs, in order to minimize credit transmission overhead and increase buffer utilization. Since only the receiver needs to use $N2$ values, it can conveniently change them locally, as described in the next section.

RECEIVER-ORIENTED ADAPTIVE BUFFER ALLOCATION

We describe the underlying idea of the receiver-oriented adaptive buffer allocation algorithm [13]. In referring to Figure 10, let *RTT* be the maximum of all the *RTT*s and M be the size, in cells, of the common buffer pool in the receiver.

For each allocation interval, which is set to be at least *RTT*, the receiver computes a new allocation and an $N2$ value for each VC according to its relative bandwidth usage. Over the allocation interval, let VU and TU be the number of cells forwarded for the VC and that for all the N active VCs, respectively. Then for the VC, the new allocation is:

$$\text{Buf\_Alloc} = (M/2 - TQ - N)^*(VU/TU) \qquad (3)$$

and the new $N2$ value is

$$N2 = \text{Buf\_Alloc}/4, \qquad (4)$$

where TQ is the total number of cells currently in use in the common buffer pool at the receiver. For exposition purposes this section ignores floor and ceiling notations for certain quantities, such as those in the right-hand sides of the above two equations. See [13] for precise definitions and analysis of all quantities.

It is easy to see that the adaptive formula of Equation (3) will not introduce cell loss. The equation says that for each allocation interval, the VCs divide a buffer of size *M*/2 - *TQ* - *N* according to their current relative bandwidth usage *VU/TU*. Thus, the total allocation for all the VCs is no more than (*M*/2 - *TQ* - *N*) + *N* or *M*/2 - *TQ*, assuming that each of the *N* VCs is always given at least one cell in its allocation. Since allocation intervals are at least *RTT* apart, after each new allocation, the total number of in-flight cells is bounded by the total *previous* allocation. Note that the total previous allocation is no more than $M/2 - TQ_{prev} \le M/2$, where TQ_{prev} is the *TQ* value used therein. Therefore the total memory usage will never exceed (*M*/2 - *TQ*) + *M*/2 + *TQ* or *M*. Consequently, adaptive buffer allocation will not cause cell loss. This analysis also explains why M is divided by 2 in Equation (3).

Equation (4) allows the frequency of transmitting credit cells of the VC, i.e., the *N*2 value, to adapt to the VC's current Buf_Alloc, or equivalently, its relative bandwidth usage. That is, VCs with relatively large bandwidth usage will use large *N*2 values. This will reduce their bandwidth overhead in transmitting credit records upstream. (In fact, by adapting *N*2 values and by packing up to 6 credits in each transmitted credit cell, the transmission overhead for credit cells can be kept very low. Simulation results in [13] show that this overhead is generally below a few percent, and sometimes below 1 percent.) On the other hand, an inactive VC could be given an *N*2 value as small as 1. With a smaller *N*2 value, the receiver can inform the sender about the availability of buffer space sooner, and thus increase memory utilization. The *N*2 value would increase only when the VC's bandwidth ramps up. Thus the required memory for each VC could be as small as one cell.

From Equations (2), (3), and (4), we can show that the adaptive scheme guarantees that a VC will ramp up to its fair share. A sufficient condition is that a fair scheduling policy is employed, the switch buffer size

$$M = 4*RTT + 2*N \qquad (5)$$

or larger is used, and a significant portion of the switch buffer is not occupied, e.g.,

$$TQ < 2*RTT/3. \qquad 6)$$

The condition of Equation (6) holds if those VCs that are blocked downstream do not occupy much buffer space at the current node. The adaptive buffer allocation scheme is indeed designed in such a way that inactive or slow VCs will be allocated the very minimum or a small buffer space, respectively.

Assume that there are *N* - 1 active VCs that in aggregate already get the full link bandwidth of an output port of the receiver. Now a new VC using the same output port starts and wishes to get its fair share, i.e., 1/*N*, of the link bandwidth. Suppose that the VC's current buffer allocation X is insufficient for achieving this target bandwidth. That is, by Equations (2) and (4),

$$\frac{X}{RTT + \frac{X}{4} \cdot N} < \frac{1}{N}$$

or, equivalently,

$$X < \frac{4 \cdot RTT}{3 \cdot N}.$$

Note that with the current allocation *X*, by Equation (2) the relative bandwidth that the VC can achieve satisfies:

$$\frac{VU}{TU} \geq \frac{X}{RTT + \frac{X}{4} \cdot N}.$$

Since TQ < 2*RTT/3, it follows from Equation (5) and the last two inequalities above that:

$$\left(\frac{M}{2} - TQ - N\right) \cdot \frac{VU}{TU} \geq (2 \cdot RTT - TQ) \frac{X}{RTT + \frac{X}{4} \cdot N} > X.$$

Thus the new allocation for the VC computed by Equation (3) will be strictly larger than *X*. In this way the buffer allocation for the VC will keep increasing after each round of new allocation, as long as the achievable bandwidth allowed by the current Buf_Alloc *X* is less than 1/*N* and the total queue length *TQ* is less than 2**RTT* /3.

In fact, the ramp up rate for a VC is exponential in number of allocations initially, when the bandwidth allowed by the credit control is small and when *TQ* is small. We can easily explain this exponential ramp up, using the last inequality expression above, for the simplifying case that *TQ* = 0. When *RTT* is large and *X**N/4 is much smaller than *RTT*, the middle term is about a factor-of-two larger than the third term. That is, *X* is ramped up roughly by a factor of two for every new allocation. In general, from the inequality expression we see that if *M* = 2* **RTT* + 2**N*, then the ramp up factor for each allocation is about . Therefore, the larger or *M* is, the faster the ramp up will be.

RATIONALE FOR CREDIT-BASED FLOW CONTROL

We discuss some key reasons behind the credit-based approach to flow control. The same rationale, perhaps formulated in a different form, is applicable to any flow control scheme.

Overallocation of Resources to Achieve High Efficiency

For reasons of efficiency, the size *M* of the *total* allocated buffer in the receiver generally needs to be larger than *RTT*. This is overallocation in the sense that if traffic is 100 percent steady state, *M* need only be *RTT* for sustaining the peak bandwidth of the output link. However, for bursty traffic, *M* needs to be larger than *RTT* to allow high link utilization and reduce transmission time.

First consider static credit control. If the required memory cost is affordable, we can let Buf_Alloc be *RTT* + *N*2 for every one of the *N* active VCs. Then by Equation (2) the maximum bandwidth the VC can achieve is at least 1/*N* for any value of *N*. When a scheduling slot for the output link becomes available, an "eligible" VC at the sender that has data and credit can transmit *instantly* at the peak link rate. When there are no other competing VCs, i.e., when *N* =

1, any single VC can sustain the peak link rate by Equation (2). Thus, link utilization is maximized and transmission time is minimized.

Now consider adaptive credit control. As in the static case, *M* needs to be large for increased link utilization and reduced transmission time. For adaptive buffer allocation, *M* needs to be large also for fast ramp up [15] as explained above.

Intuitively, receiver-oriented adaptation needs more buffer than does sender-oriented adaptation, because receiver-oriented adaptation involves an extra round-trip delay for the receiver to inform the sender of the new allocation. Thus the minimum buffer size for receiver-oriented adaptation is increased from *RTT* to 2**RTT*. Suppose that the total memory size is larger than the minimum 2**RTT*, e.g., as given by Equation (5). Then the part of the memory that is above the minimum 2**RTT* will provide "headroom" for each VC to increase its bandwidth usage under the current buffer allocation. If the VC does increase its bandwidth usage, then the adaptation scheme will notice the increased usage and will subsequently increase the buffer allocation for the VC [12].

The receiver-oriented adaptive buffer allocation scheme in [13] uses *M* given by Equation (5). Analysis and simulation results have shown that with this choice of *M* the adaptive scheme gives good performance in utilization, fairness, and ramp up [13].

Link-By-Link Flow Control to Increase Quality of Control

Link-by-link flow control has shorter and more predictable control loop delay than does end-to-end flow control. This implies smaller memory requirements for switching nodes and higher performance in utilization, transmission time, fairness, and so on.

Link-by-link flow control is especially effective for handling transient "cross" traffic. Consider Figure 11, where *T* is an end-to-end flow-controlled traffic using some end-to-end transport-level protocol such as TCP and *X* is high-priority cross traffic. If *X* uses the whole bandwidth of the Switch3's output link, then the entire window of *T* for coveting the end-to-end round-trip delay would have to be buffered to avoid cell loss. With link-by-link flow control, all the buffers on the path from the source of *T* to Switch3 can be used to prevent cell loss. In contrast, without link-by-link flow control, only the buffer at the congestion point (i.e., Switch3 in this case) can be used for this purpose. The argument for making efficient use of buffers is similar to that for making efficient use of dams in the flood-control analogy described above.

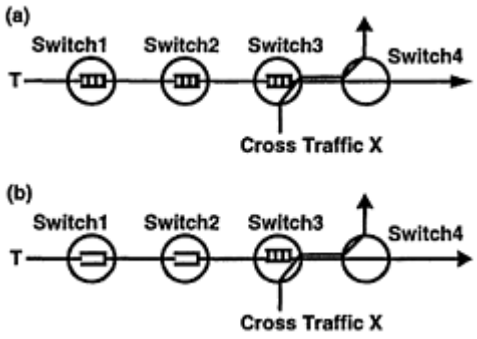

Figure 11 (a) With link-to-link flow control, all buffers on the path leading to the congestion point (Switch3) where traffic T meets cross traffic X can be used for preventing cell loss; (b) without link-by-link flow control, only the buffer in Switch3 can help.

Moreover, sufficient predictability in the control loop delay is necessary for the receiver to perform policing. After issuing a flow control command to the sender, the receiver will need to start policing the traffic according to the new condition only after the control loop delay. If control loop delay cannot be bounded, it is impossible for the receiver to decide when to start policing.

Per-VC Queueing to Achieve a High Degree of Fairness

To achieve fairness between bursty VCs sharing the same output, it is necessary to have separate queueing for individual VCs. With a fair round-robin scheduling policy among these queues, cells from different VCs will be sent out in a fair manner.

Per-VC queueing provides firewall protection against VCs interacting each other. Technology advances have lowered the cost of implementing per-VC queueing. There are more and more per-VC queueing switches available on the market. Fore System's ASX200WG is one example. Per-VC queueing will be the future trend for ATM technology.

Rate-Based Flow Control

It is instructive to consider rate-based flow control schemes [3, 4], in contrast to the credit-based approach described above. Rate-based flow control consists of two phases: *rate setting* by sources and network, and *rate control* by sources. These two phases correspond to the buffer allocation and credit control phases in credit-based flow control.

Rate control is a shaping function for which various implementations are possible. For example, when a cell of a VC with a given rate r arrives, the cell will be scheduled for output at time $1/r$ after the previous output of the same VC. By sorting arriving cells into buckets according to their departure times, rate control can be implemented without per-VC queueing (although per-rate-bucket queueing may be needed).

Suppose that traffic is so smooth that it is possible to set the rate for each VC perfectly against some performance criteria, and that these rates need not change over time to sustain the target performance. Then, if the VCs are shaped at the sources according to the set rates, the rate-based flow control method should work perfectly well. There would be no need for link-by-link flow control and per-VC queueing in the network. The buffer in a switch could also be kept at the minimum, almost as in a synchronous transfer mode (STM) switch.

However, setting rates perfectly or near optimally is a complicated matter. Consider, for example, the configuration in Figure 12, known at the Traffic Management Group of the ATM Forum in 1994 as Generic Fairness Configuration (GFC) [20]. All traffic sources are assumed to be persistently greedy and can transmit at the peak link rate when bandwidth is available. Links

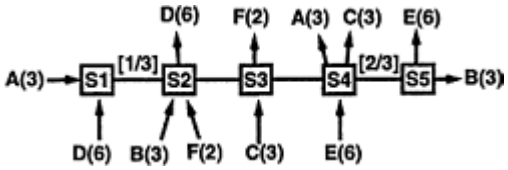

Figure 12 Generic Fairness Configuration (GFC). [ ]: Link bandwidth; link bandwidth = 1 if not indicated. (k): Number of virtual circuits (VCs) in the VC group.

have various propagation delays. The actual values of the propagation delays are not important to the discussion here, and thus are not listed.

Note that both traffic B and E share the same link between S4 and S5, and the source of E is closer to the link than that of B. This is analogous to a parking lot scenario in which E starts from a position closer to the exit than B. In a normal, real-world parking lot, E would have an unfair advantage over B by being able to move itself in front of B and get out first. However, in a good ATM network with separate virtual circuits for B and E, they ought to share fairly the bandwidth of the link, as long as they are not bottlenecked elsewhere in the network.

With this fairness objective in mind, we naturally consider the performance criterion described below, which is sometimes called "max-min" fairness [3, 4, 6] in the literature. First, the VCs on the most congested link will share the link bandwidth equally, and this determines the rates to be set for these VCs. Then, apply the procedure to the other VCs with the remaining bandwidth of the network. Continue repeating the procedure until rates for all the VCs have been assigned. Table 1 shows the resulting rates assigned to individual VC groups.

Translating the above mathematical rate-setting procedure into an efficient and robust implementation is a major challenge. First, with highly bursty ABR traffic, because load changes rapidly, there would be no static rate-setting that could be ideal for any significant period of time. When traffic changes, "optimal" rates to be assigned to the affected VCs must change accordingly.

For this reason, adaptive rate-setting is necessary for bursty traffic and has been the subject of intensive research for many years. The Enhanced Proportional Rate-Control Algorithm (EPRCA) [18], one of the schemes considered at the 1994 ATM Forum, represents the kind of adaptive rate-setting schemes this paper assumes.

Rate adaptation *cannot* be so precise that the newly derived rates will be exactly right with respect to current load, for at least two reasons. First, information and measurements based on which particular adaptation is performed cannot be totally complete or up to date due to various cost and implementation constraints. Second, the feedback control time that the adaptation takes to inform sources can vary because of disparities in propagation delay and link speed, congestion conditions, scheduling policies, and many other factors.

More interesting, perhaps, is that rate adaptation *should not* be precise either. To achieve high utilization with bursty traffic, it is necessary that the total assigned rate for all the VCs over a link be higher than the peak link rate. Consider the simple scenario shown in Figure 13 involving only two VCs, A and B. Assume that the two VCs share the same switch output link

TABLE 1 Expected Rates for VC Groups in Generic Fairness Configuration (GFC) of Figure 12

Group	Bandwidth	Bottleneck Link
A	1/27 = 0.037	S1-S2
B	2/27 = 0.074	S4-S5
C	2/9 = 0.222	S3-S4
D	1/27 = 0.037	S1-S2
E	2/27 = 0.074	S4-S5
F	1/3 = 0.333	S2-S3

of bandwidth 1, and that each has a data burst that would take a unit time to transmit over a link of bandwidth 1. Suppose that the B burst arrives 1 unit time later than the A burst. Then as Figure 13 depicts, in the precise rate-setting case where each VC is set with a rate of 0.5, it would take a total of 3 time units to complete the transmission of both the A and B bursts. In contrast, in the overallocating rate-setting case where each VC is set with a rate of 1, it would take only 2 time units to do the same. This need for overallocating resources is similar to that discussed above for credit control.

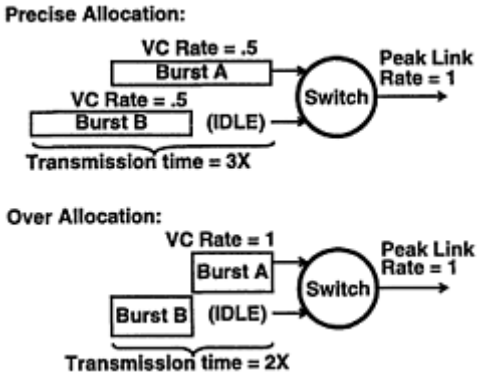

Figure 13 Both bursts A and B complete transmission earlier and make higher utilization of switch output link in the overallocating case than in the precise case.

Since adaptation *cannot* and *should not* be precise, rates set by the adaptation may not be totally correct. Bounding the liability of overrunning switch buffers is therefore a first-order issue. This explains why credit-based flow control has been desired to control buffer allocation and monitor the buffer usage, directly.

CreditNet ATM SWITCH

To study ATM-layer flow control, BNR and Harvard University have jointly developed an experimental ATM switch [2], with both 622-Mbps (OC-12) and 155-Mbps (OC-3) ports. This effort is part of the CreditNet research project supposed, in part, by the Defense Advanced Research Projects Agency (DARPA). Under this project, BNR and Hazard have developed the ATM switch described here, whereas Carnegie Mellon University (CMU) and Intel have developed an ATM-PCI host interface at both OC-3 and OC-12 rates.

This experimental CreditNet switch has a number of unique features. These include ATM-layer credit-based flow control, per-VC round-robin cell scheduling, multicast support in hardware, highly programmable microprocessor-based switch port cards, and built in instrumentation for performance measurement. (Independently, Digital Equipment Corporation (DEC) has also developed a credit-based ATM network.)

Five of these experimental switches have been built; the first one has been operational since spring 1995. Several ATM host adapters have been used in conjunction with the switch. These include those from DEC (for TurboChannel), Sun (S-Bus), Intel (PCI) and Zeitmet (PCI). Both the OC-3 and OC-12 links have been used in various experiments. In addition, a Q93B signaling

system has been successfully implemented on the switch. As of spring 1996, one of the switches now operates on site at Harvard, one is temporarily at a Sprint site to support a WAN congestion control trial, and others are at BNR and CMU.

To implement credit-based flow control, the switch monitors the buffer use of each VC and provides feedback to the immediately preceding switch or host along the VC's path. Since each switch has precise knowledge of the resources a circuit is consuming, and the feedback loop is only one link long instead of the length of the entire end-to-end connection, this flow control system allows much more efficient use of buffer memory and link bandwidth.

As shown in Figure 14, the switch is physically organized as 16 modules that plug into a backplane and memory buffer system. One of the modules is the switch control module for call processing using the Q93B signaling standard.

The rest of the modules are port modules. Each port module has two 960 microprocessors, one for scheduling mentioned above and one to handle real-time monitoring and control. These two microprocessors are not necessary for a minimum implementation of a credit-based switch, but they provide the programming flexibility necessary to study many research issues. For example, these processors provide the flexibility to experiment with different ways of observing and reacting to network load conditions. Each port module also has a fiber-optic link interface using Synchronous Optical Network (SONET) framing. The cell-handling hardware is built from field-programmable gate arrays for control, and from static random access memories (RAMs) for tables and queues.

When a cell arrives at the switch, the input port broadcasts the cell's circuit identifier and address in the common memory on the arrival bus on the backplane. Each output port monitors this backplane; when a port notices that a cell has arrived for a circuit that leaves that port, it adds the cell's memory address to a queue.

When a cell leaves an output port, its circuit identifier is broadcast on the departure bus on the backplane. By watching the arrival and departure buses, each input port maintains a count of the number of cells buffered for each circuit that enters that port. This count is used both to provide credit-based flow-control feedback and to decide which circuits are using so much memory that their data should be discarded.

The common memory architecture allows the switch to support multicast in an efficient way. A common memory allocation engine maintains a list of free locations in the shared common

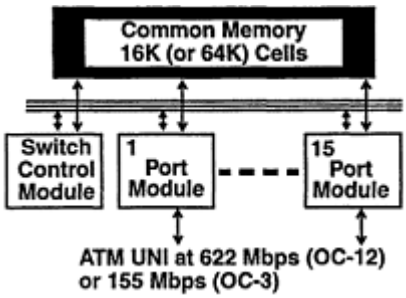

Figure 14 Architecture overview of CreditNet switch.

memory. Entries from this list are allocated to cells as they arrive. When a multicast cell's information is broadcast on the arrival bus, more than one port module will enqueue this cell in its list of cells to send. However, the cell requires only one common memory location.

The allocation engine hands out addresses of free slots in the common memory to the ports on demand, so that they can store incoming data. When it does this, it initializes a list of egress ports that must send this cell out. When a port sends out a cell, the presence of the cell's identifier on the departure bus tells the allocation engine to remove it from the list. When the list becomes empty, the common memory location is recycled for future use. All this is done by efficient hardware: the allocation engine requires only four memory accesses per port per cell cycle.

For most purposes, the ingress and egress sides of a port are effectively independent. However, they have an important interaction required for the credit protocol. Essentially, the credit protocol requires a sender to have a credit for a given VC, before sending cells on it. Credit is granted by sending credit cells opposite the flow of data (from receiver to sender). Thus, when the ingress side of a port realizes that a number of cells for that VC have left the switch, it notifies the egress side of the same port to send a credit cell.

EXPERIMENTAL NETWORK CONFIGURATIONS

The CreditNet switch has been used to experiment with TCP performance over ATM networks. The experiments described below use two network configurations in a LAN environment. The first, shown in Figure 15 (a), involves host A sending a continuous stream of data through the switch to host B. Host A's link to the switch runs at 155 Mbps, while host B's link runs at only 53 Mbps, enforced by a properly programmed scheduler on the link input. This is one of the simplest configurations in which congestion occurs. Note that after SONET and ATM overhead, a 155-Mbps link can deliver roughly 134 Mbps or 17 megabytes per second (Mbyte/sec) of useful payload to a host. A 53-Mbps link can deliver about 5.7 Mbyte/sec.

The second configuration, shown in Figure 15 (b), involves four hosts. Host A sends data to host C, and host B to host D. The four host links run at 155 Mbps, and the bottleneck link between the switches runs at 53 Mbps. The purpose of this configuration is to show how two conversations interact.

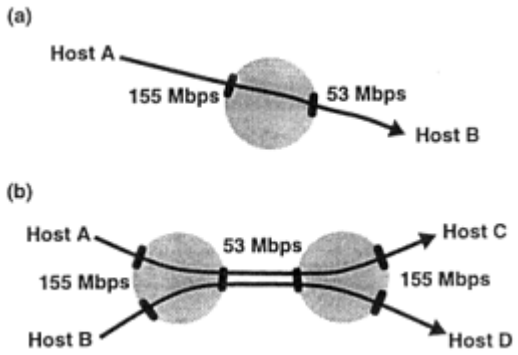

Figure 15 (a) Network configuration for single TCP experiments on CreditNet and (b) configuration for two competing TCPs. The shaded circles represent switches. Each darkened bar denotes a switch port.

The hosts in all these experiments are DEC Alpha 3000/400 workstations running OSF/1 V3.0. The OSF/1 TCP implementation [5], used in all the experiments reported in this paper, is derived from 4.3-Reno [21]. This TCP tends to acknowledge, and thus transmit, pairs of packets. The TCP window size for these experiments is limited to no more than 64 kbytes, and the packet size to 9180, except when noted. The workstations use 155-Mbps OTTO TurboChannel adapters provided by DEC. The Alphas can send or receive TCP using the OTTOs at about 15 Mbyte/sec. The OTTO drivers optionally implement CreditNet's credit-based flow control partially in software; with credit turned on they can send and receive TCP at 13 Mbyte/sec.

The measurements are all directly derived from the instrumentation counters in the CreditNet switch hardware. The hardware keeps track of the total number of cells sent by each VC and the number of cells buffered for each VC.

MEASURED PERFORMANCE ON CREDITNET EXPERIMENTAL SWITCHES

ATM-layer credit-based flow control resolves some TCP performance problems over ATM networks when packets are lost because of congestion [16]. The bottleneck switch no longer discards data when it runs out of buffer memory and possibly causes TCP to endure lengthy time-out periods. Instead, it withholds credit from the switches and/or hosts upstream from it, causing them to buffer data rather than sending it. This backpressure can extend all the way back through a network of switches to the sending host. The effect is that a congested switch can force excess data to be buffered in all the upstream switches and in the source host. Data need never be lost because of switch buffer overran. Thus, if TCP chooses a window that is too large, the data will simply be buffered in the switches and in the host; no data loss and retransmission time-outs will result.

Table 2 compares the useful bandwidths achieved with and without credit-based ATM-layer flow control in the configurations shown in Figure 15. For the flow-controlled cases, the switch has 100 cell buffers (4800 payload bytes) reserved per-VC. For the non-flow-controlled cases, the switch has 682 (32 payload kbytes) cells of buffering per-VC. Recall that for the configuration in Figure 15 the slow link can deliver at most 5.7 payload Mbps, and the fast link 17. Thus in both the one TCP and two TCPs cases, TCP with credit-based flow control achieves its maximum-possible bandwidth.

Using a configuration similar to that shown in Figure 15 (b), experiments involving one TCP and one UDP, instead of two TCPs, have also been carded out. A typical measured result is as follows. When ATM-layer credit-based flow control is used, UDP gets its maximum bandwidth limited only by the source, while TCP gets essentially the remaining bandwidth of the bottleneck link between the two switches. However, when credit-based flow control is turned off, TCP's throughput drops significantly and the total utilization on the bottleneck link by both TCP and UDP is reduced to less than 45 percent. Thus, when competing with UDP, TCP with ATM-layer flow control can keep up its throughput even though UDP does not reduce its bandwidth during network congestion.

TABLE 2 Measured Total Bandwidth Achieved with and without ATM-Layer Credit-based Flow Control, for the (a) and (b) configurations of Figure 15

	Without ATM-Layer Flow Control	With ATM-Layer Flow Control
(a) One TCP	0.1 Mbyte/sec	5.7 Mbyte/sec
(b) Two TCPs	0.2 Mbyte/sec	5.7 Mbyte/sec

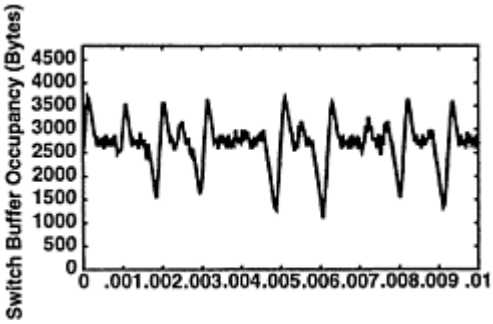

Figure 16 Measured switch buffer occupancy when one TCP scuds into a slow link, as depicted in Figure 15 (a), with credit flow control turned on.

Figure 16 shows how much switch buffer space is used when one TCP sends into a slow link with credit flow control turned on, for the configuration depicted in Figure 15 (a). The flow control system makes sure that enough cells are always buffered that it can keep the output link busy, but never much more than that. The large oscillations correspond to packet boundaries.

Summary and Concluding Remarks

Although the cost of memory has been dropping over the years, the speed of networks and the potential number of simultaneous applications sharing a network have also been increasing. The brute-force way of simply enlarging buffers to avoid data loss will quickly become technically and economically impractical.

Traffic management is therefore essential. For data applications, we need to ensure no loss due to congestion, high utilization, and fairness, regardless of traffic patterns. This kind of guarantee may be a requirement, not just a luxury, in order to provide acceptable service under harsh conditions expected in real-world data traffic.

The most visible sign of network overload due to traffic bursts is usually buffer exhaustion. Credit flow control directly controls buffer allocation and monitors its usage.

Analysis, simulation, and actual experiments on switching hardware have shown that credit flow control can work well over a wide range of network conditions. At one extreme, static allocation of buffers is simple and provides the guarantee. The adaptive credit flow control system can reduce memory requirements to just a few round-trip times' worth of cells, while maintaining no loss and high performance. Thus, a credit system can provide good performance even if future networks are nothing like those currently predicted. Credit flow control is an existence proof that control of congestion can enforce a guarantee of no data loss.

As our field experience with ATM networks expands, we will have much to learn, especially about the interaction of ATM flow control with higher-lever protocols. Future research in congestion control should explore the patterns of real traffic on high-speed networks. Working prototypes of the competing flow control systems should be compared. Without such experience it is not possible to make proper trade-offs between performance and cost.

Acknowledgments

This research was supported in part by corporations including Intel, Nortel, and Ascom Nexion, and in part by the Defense Advanced Research Projects Agency (of the DOD) monitored by DARPA/CMO under Contract MDA972-90-C-0035 and by AFMC under Contract F1VP9628-92-C-0116. Parts of, the paper are excerpts from earlier publications [11, 15, 16] by the author and his co-authors. QuickTime movies capturing various experimental results of, the CreditNet switch can be accessed at http://www.eecs.harvard.edu/cn-traces.html.

References

[1] ATM Forum, "ATM User-Network Interface Specification," Version 3.0, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] T. Blackwell et el., "An Experimental Flow Controlled Multicast ATM Switch," *Proceedings of the First Annual Conference on Telecommunications R&D in Massachusetts*, Vol. 6, pp. 33-38, October 25, 1994.

[3] A. Charny, "An Algorithm for Rate Allocation in a Packet-Switching Network with Feedback," MIT/LCS/TR-601, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, April 1994.

[4] A. Charny, D. Clark, and R. Jain, "Congestion Control with Explicit Rate Indication," *Proceedings ICC'95*, p. 10, June 1995.

[5] Chran-Ham Chang et el., "High-performance TCP/IP and UDP/IP Networking in DEC OSF/1 for Alpha AXP," *Digital Technical Journal*, Winter 1993.

[6] E.L. Hahne, "Round-Robin Scheduling for Max-Min Fairness in Data Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 7, pp. 1024-1039, September 1991.

[7] G. Irlam, "Unix File Size Survey—1993," Usenet comp.arch.storage, <URL: http://www.base.com/gordoni/ufs93.html>, last updated September 1994.

[8] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson, "On the Self-Similar Nature of Ethernet Traffic," *Proceedings of the ACM SIGCOMM 1993 Symposium on Communications Architectures, Protocols, and Applications*, pp. 183-193, September 1993.

[9] M.G.H. Katevenis, "Fast Switching and Fair Control of Congested Flow in Broadband Networks," *IEEE Journal on Selected Areas in Communications* , Vol. SAC-5, No. 8, pp. 1315-1326, October 1987.

[10] V. Jacobson, "Congestion Avoidance and Control," *Proceedings of the SIGCOMM 1988 Symposium on Communications Architectures and Protocols*, August 1988.

[11] H.T. Kung, "Flow-Controlled ATM Switches for Available Bit Rate Services", *Proceedings of the 2nd International Conference on Massively Parallel Processing Using Optical Interconnections*, pp. 176-179, IEEE Computer Society Press, San Antonio, Texas, October 1995.

[12] H.T. Kung, T. Blackwell, and A. Chapman, "Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation, and Statistical Multiplexing," *Proceedings of the ACM SIGCOMM 1994 Symposium on Communications Architectures, Protocols, and Applications*, pp. 101-114, August 31-September 2, 1994.

[13] H.T. Kung and K. Chang, "Receiver-Oriented Adaptive Buffer Allocation in Credit-Based Flow Control for ATM Networks," *Proceedings of INFOCOM 1995*, pp. 239-252, April 1995.

[14] H.T. Kung and A. Chapman, "The FCVC (Flow-Controlled Virtual Channels) Proposal for ATM Networks," Version 2.0, 1993. A summary appears in *Proceedings of the 1993 International Conference on Network Protocols*, pp. 116-127, San Francisco, October 19-22, 1993. (PostScript files of this and other related papers by the authors and their colleagues are available via anonymous FTP from virtual.harvard.edu:/pub/htk/atm.)
[15] H.T. Kung and R. Morals, "Credit-Based Flow Control for ATM Networks," *IEEE Network Magazine*, pp. 40-48, March/April 1995.
[16] R. Morris and H.T. Kung, "Impact of ATM Flow Control on TCP Performance: Measurements on an Experimental ATM Switch," *Proceedings of GLOBECOM 1995*, pp. 888-892, Stamford Press, Singapore, 1995.
[17] C. Ozveren, R. Simcoe, and G. Varghese, "Reliable and Efficient Hop-by-Hop Flow Control," *Proceedings of the ACM SIGCOMM 1994 Symposium on Communications Architectures, Protocols, and Applications*, pp. 89-100, August 31-September 2, 1994.
[18] Larry Roberts, "Enhanced PRCA (Proportional Rate-Control Algorithm)," ATM-Forum/94-0735R1, August 1994.
[19] A. Schmidt and R. Campbell, "Internet Protocol Traffic Analysis with Applications for ATM Switch Design," *ACM SIGCOMM Computer Communication Review*, Vol. 23, No, 2, pp. 39-52, April 1993.
[20] R.J. Simcoe, "Configurations for Fairness and Other Tests," ATM Forum/94-0557, 1994.
[21] G. Wright and W.R. Stevens, *TCP/IP Illustrated*, Vol. 2, Addison-Wesley, New York, 1995.