This article is to appear in
IEEE Network Magazine, March 1995.

Credit-Based Flow Control for ATM Networks

H. T. Kung and Robert Morris
Division of Applied Sciences, Harvard University, 29 Oxford Street, Cambridge, MA 02138
Email: kung@das.harvard.edu, rtm@das.harvard.edu

Abstract

Congestion control ensures that network resources are divided fairly and efficiently among competing connections. Although congestion control has been studied by researchers for a long time, very high-speed networks using ATM pose a new set of challenges.

The proposed credit-based mechanism provides flow control tailored to ATM networks. Simulation, analysis and experiments on switching hardware have shown that for a wide variety of traffic patterns, credit control is fair, uses links efficiently, minimizes delay, and guarantees no cell loss due to congestion. The credit system is especially well suited to data traffic that is bursty, unpredictable, and has little tolerance for delay.

Other approaches to flow control, including rate-based flow control, may require less expensive hardware and may be effective for steady traffic, but do not handle bursty traffic well. While no one can predict what kind of traffic will dominate future networks, recent evidence suggests that it will be bursty. Thus a major challenge for network research will be to find congestion control mechanisms that blend the hardware simplicity of rate-based flow control with credit-based flow control's ability to handle bursts. This research, in turn, will depend on more experience with real applications and high-level protocols running over ATM.

The high-speed networking market is solidifying quickly, and basic issues in congestion control should be well understood before the market is forced to choose a solution with unknown properties. The lessons learned from the credit-based approach should be incorporated into whatever flow control system that is finally standardized or widely implemented.

This research was supported in part by BNR and Intel, and in part by the Advanced Research Projects Agency (DOD) monitored by ARPA/CMO under Contract MDA972-90-C-0035 and by AFMC under Contract F19628-92-C-0116.

1. Introduction

Over the last year the ATM standards community has recognized that data traffic often requires no firm guarantee of bandwidth, but instead can send at whatever rate is convenient for the network. This is called "Available Bit-Rate" traffic by the ATM Forum. ABR traffic gives the network the opportunity to offer guarantees to high priority traffic, and divide the remaining bandwidth among ABR connections.

To support ABR traffic the network requires a feedback mechanism in order to tell each source how much data to send. A number of such mechanisms have been proposed for ATM, with considerable debate as a result. The two main mechanisms are called credit-based flow control and rate-based flow control. In late 1994, the ATM Forum voted for rate-based flow control, but without committing to the details of any particular algorithm.

This article summarizes the technical basis for credit flow control, including some fundamental advantages of credit which could be adopted by other mechanisms. We hope thereby to speed the evolution of ATM flow control, and minimize the risk of standardizing inadequate solutions. This article avoids political and short-term pragmatic issues, such as migration paths and interoperability, noting that flow control mechanisms adopted now may be in use long after such issues are forgotten.

Section 2 introduces the problem that flow control solves, along with desirable properties of any solution. Section 3 presents two general kinds of network traffic that flow control must cope with. Section 4 provides details of the credit mechanism, using static or dynamic buffer allocation, and Section 5 gives some intuition as to why it works well. Section 6 outlines the rate-based scheme. Section 7 describes how to add benefits of credit control to rate-based systems. Section 8 summarizes some flow control simulation studies and actual experimental results on ATM switching hardware. Section 9 contains fundamental reasons why credit flow-control has advantages over rate-based, and the concluding section suggests ways in which the advantages of both may be combined.

2. Flow control problem

Any data network has bottlenecks: points where more data can arrive than the network can carry. These points are often in switches with multiple ports; congestion arises when data, destined for a single output port, arrives at many inputs. The universal short-term solution involves buffer memory in which a switch can temporarily queue data directed at overloaded outputs. In the longer term, no amount of buffering is sufficient: 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.

This is fundamentally a feedback control problem, and many control ideas and principles apply. Each network switch 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 switch. The sources act to control how much data they send. This control loop has a delay of at least twice the propagation delay between the switch and control point. Control systems should seek to minimize this delay, since switches will need to buffer any data that arrives after they signal the congestion status but before the end of the delay.

A variety of technical goals, some of them conflicting. are desirable for any flow control mechanism. Data should rarely, if ever, be discarded due to exhaustion of switch buffer memory. Such data may have to be retransmitted after a possibly lengthy time-out period, further contributing to network congestion and the delay seen by the user. Links between switches 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 as soon as possible. All the connections which 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, for instance, should not cause increased congestion. The network administrator should not have to adjust any complex parameters to achieve high performance. Finally, the flow control mechanism should have a cost commensurate with the benefits it provides.

3. 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 purposes it is enough to distinguish between *smooth* and *bursty* traffic.

A smooth traffic source offers a constant and predictable load, or only changes in 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. 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 wide-area network where the individual sources are relatively low-bandwidth and uncorrelated. Rate-based flow control works well with smooth traffic.

Bursty traffic lacks any of the predictability of smooth traffic, as observed in some computer communications traffic [5]. Some kinds of bursts stem from users and applications. A Mosaic user 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 Mosaic'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.

The most visible sign of network overload due to traffic bursts is usually buffer exhaustion. Credit flow control works well with such traffic because it directly controls buffer allocation.

4. Credit-based flow control

We briefly review credit-based flow control. Implementing link-by-link, per-VC (virtual circuit) flow control, the scheme generally works over a VC link as follows. As depicted by Figure 1, 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 decrements its current credit balance for the VC by one.

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

An experimental OC-12 (622-Mbps) ATM switch [2] with credit flow control has been developed by BNR and Harvard. An ATM network testbed involving multiple

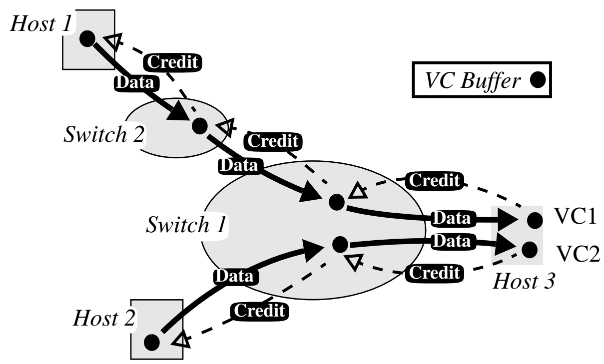

Figure 1: Credit-based flow control applied to each link of a VC

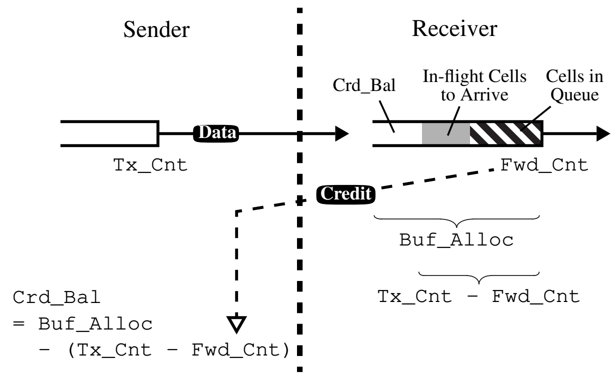

Figure 2: Credit Update Protocol (CUP)

copies of this switch and a variety of ATM host adapter cards is operational. Experiments on this testbed have confirmed the benefit of credit-based flow control as described in this article. (See Section 8.) Independently, Digital Equipment Corporation has also developed a credit-based ATM network.

For the remaining section, we first describe a protocol for the credit control phase. Then we introduce the notion of static vs. adaptive credit control, reflecting if the buffer allocation is static or adaptive. Next, we overview the sender- and receiver-oriented adaptive approaches. Finally, we describe in some detail a receiver-oriented adaptive buffer allocation scheme.

4.1. Credit Update Protocol

The *Credit Update Protocol* (CUP) [8] is an efficient and robust protocol for implementing credit control over a link. (A link can be a physical link connecting two adjacent nodes, or a virtual circuit connecting two remote nodes.) As depicted by Figure 2, 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 are 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:

`Crd_Bal = Buf_Alloc – (Tx_Cnt – Fwd_Cnt)` (1)

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

Note that the quantity, `Tx_Cnt – Fwd_Cnt`, represents the "outstanding credits" which correspond those cells of the VC which the sender has transmitted but the receiver has not forwarded. As depicted in Figure 2, 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 sender transmits no more than `Crd_Bal` cells, it will not overrun the VC's allocated buffer in the receiver. See [8] 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.

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 forwarded "`N2`" cells for some positive integer `N2`, the receiver will send a credit record upstream. The value of `N2` can be set statically or adaptively.

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 Section 4 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 times, of the link between the sender and the receiver (see Figure 2) 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 = Buf_Alloc / (RTT + N2*N)` (2)

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

The CUP scheme is a lower level and lighter weight protocol than 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 advancing the 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.

It can be shown [10] that CUP produces the same buffer management results as the well-known "incremental" credit updating methods (see, e.g., [3, 6]). 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.

4.2. Static vs. adaptive credit control

We call a credit-based flow control *static* or *adaptive*, if the buffer allocation is static or adaptive, respectively. In a static credit control, a fixed value of `Buf_Alloc` will be used for the lifetime of a VC. Requiring only the implementation of CUP in Section 4.1 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 get sufficient data and 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 may be used to adjust `Buf_Alloc` adaptively. The adaptation can be carried out by software.

4.3. 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 will automatically decrease, 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 will automatically be assigned to other VCs which have data to forward and are not congested downstream.

Adaptive buffer allocation can be implemented at the sender or receiver node. As depicted by Figure 3, in a sender-oriented adaptive scheme [8, 11] 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* [8].

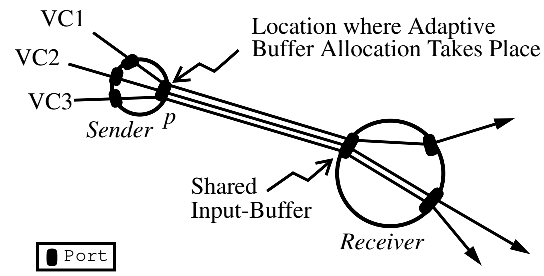

Figure 3: Sender-oriented adaptation

Receiver-oriented adaptation [9] is depicted by Figure 4. 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* [9].

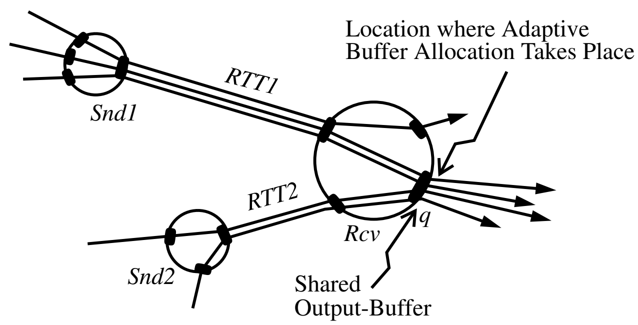

Figure 4: Receiver-oriented adaptation

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 4 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*) can observe the bandwidth usage of the VCs from *all* the senders (that is, *Snd1* and *Snd2* for Figure 4). In contrast, each sender can only observe the bandwidth usage 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 Section 4.4.

4.4. Receiver-oriented adaptive buffer allocation

We describe the underlying idea of the receiver-oriented adaptive buffer allocation algorithm in [9]. In referring to Figure 4, let RTT be the maximum of all the RTTs 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 will compute a new allocation and an N2 value for each VC according its relative bandwidth usage. Over the allocation interval, let VU and TU be the number of cells forwarded for the VC and that forwarded for all the N active VCs, respectively. Then for the VC, the new allocation is:

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

and the new N2 value is:

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

where TQ is the total number of cells currently in use in the common buffer pool at the receiver. For the purpose of presenting the basic adaptive idea here, it is without loss of generality that in this section floor and ceiling notations for certain quantities are ignored, such as those in the right-hand sides of the above two equations. See [9] 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 the "pie" 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 the 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 $M/2 - TQ_{prev} \leq M/2$, where TQ_{prev} is the TQ value used then. Therefore the total memory usage will never exceed (M/2 - TQ) + M/2 + TQ or M, and 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 N2 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 N2 values, and thus will reduce their bandwidth overhead of transmitting credit records upstream. (In fact, by adapting N2 value and by packing up to 6 credit credits in each transmitted credit cell, the transmission overhead for credit cells can be kept very low. Simulation results in [9] 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 N2 value as small as one. With a smaller N2 value, the receiver can inform the sender the availability of buffer space sooner, and thus increase memory utilization. The N2 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

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

or larger is used, and a significant portion of the switch buffer is not occupied, i.e., TQ < 2*RTT/3.

Assume that there are N - 1 active VCs which 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 two every new allocation. In general, from the inequality expression we see that if M = 2*a*RTT+2*N, then the ramp up factor for each allocation is about α. Therefore the larger α or M is the faster the ramp up is.

5. Rationale

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

5.1. Resource over-allocation to achieve high efficiency

For efficiency reasons, the size M of the *total* allocated buffer in the receiver generally needs to be larger than RTT. This is over-allocation 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, we need M to be larger than RTT to allow high link utilization and reduce transmission time.

First consider static credit control. If it is affordable, we can let Buf_Alloc be RTT+N2 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., 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, as analyzed in the end of Section 4.

Intuitively, receiver-oriented adaptation needs RTT more buffer than 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 grow its bandwidth usage under the current buffer allocation. If the VC does increase its bandwidth usage, then as described in Section 4.3 the adaptation scheme will notice the increased usage and will subsequently increase the buffer allocation for the VC [8, 9].

The receiver-oriented adaptive buffer allocation scheme in [9] uses M given by Equation (5). Analysis and simulation results have shown that with this choice of M the adaptive scheme gives excellent performance in utilization, fairness, and ramp-up [9].

5.2. Link-by-link flow control to increase quality of control

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

Link-by-link flow control is especially effective for handling transient "cross" traffic. Consider Figure 5 where T is an end-to-end flow controlled traffic using some 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 covering 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.

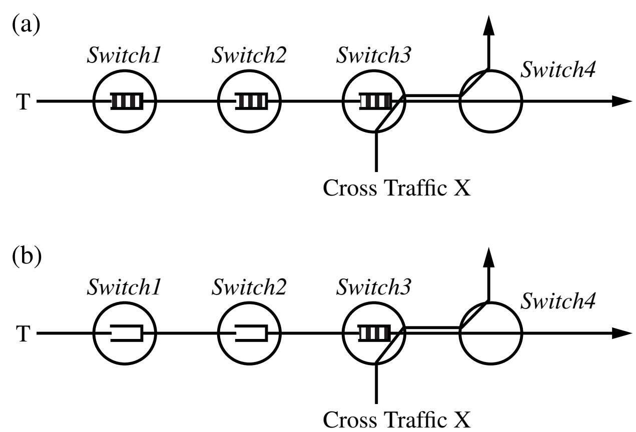

Figure 5: (a) With link-by-link flow control, all buffers on the path leading to the congestion point (i.e., Switch3) where traffic T meets cross traffic X can be used for preventing cell loss; (b) without link-by-link flow control, only the buffer in Switch3 can be used.

Moreover, sufficient predictability in the control loop delay is necessary for the receiver is to perform policing. After issuing a flow control command to the sender, the receiver will start policing the traffic according to the new condition only after the control loop delay. Effective policing will not be possible if control loop delay cannot be bounded.

5.3. Per-VC queueing to achieve high degree of fairness

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

6. Rate-based flow control

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 steady-state that it is possible to set the rate for each VC perfectly against some performance criteria, and 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 just fine. There would be no need for link-by-link flow control and per-VC queueing in the network. The buffer in a switch can also be kept at the minimum, almost like in a synchronous transfer mode (STM) switch.

However, setting rates perfectly or near optimally is a complicated matter. Consider, for example, the configuration of Figure 6, known at the ATM Forum as "Generic Fairness Configuration" (GFC) [14]. All traffic sources are assumed to be persistently greedy, and can transmit at the peak link rate when the bandwidth is available.

PD: Propagation delay in cell transmission times over a link of bandwidth 1
PD = 1 for link between host and switch
[ ]: Link bandwidth
Link bandwidth = 1 if not indicated
(k): Number of VCs in the VC group

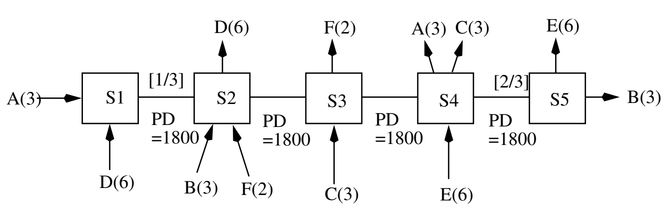

Figure 6: Generic Fairness Configuration (GFC)

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. 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.

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

Table 1: Expected rates for VC groups in generic fairness configuration (GFC)

Translating the above mathematical rate-setting procedure into an efficient and robust implementation is a major challenge. First, under highly bursty ABR traffic, because load changes rapidly, there would be no static rate setting that can 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 subject to intensive research for many years. The "Enhanced Proportional Rate-Control Algorithm (EPRCA)" [12] is one of the latest schemes considered at the ATM Forum, and represents the kind of adaptive rate setting schemes this article assumes.

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

Rate adaptation should not be precise either. To achieve high utilization under 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 a simple scenario of Figure 7 involving only two VCs, A and B. Assume that the two VCs share the same switch output link of bandwidth 1, and each have a data burst that would take a unit time to transmit over a link of bandwidth 1. Suppose that the B burst arrives one unit time later than the A burst. Then as Figure 7 depicts, in the precise rate setting case where each VC is set with a rate of .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 over-allocating 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 of over-allocating resources is similar to that discussed in Section 5.1 for credit control.

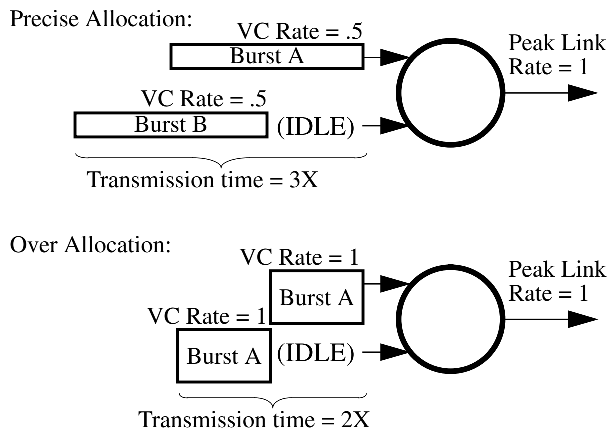

Figure 7: Both bursts A and B complete transmission earlier and make higher utilization of switch output link in the over-allocating case than in the precise case.

As discussed above, since adaptation *can not* and *should not* be precise, rates set by the adaptation may not be totally correct. Bounding the liability of overrunning switch buffers is then a first order issue. Ideas from credit control's method for preventing buffer overflow may be exploited. This is the subject of the next section.

7. Rate-based control in credit style

Rate control can use the credit style of buffer overflow prevention if the sender reduces its transmission rate when the receiver's buffer gets to be full. In credit control, the receiver in this case will send upstream reduced credit and allocation amounts, as described in Section 4. In rate control, the receiver can similarly send upstream reduced rates, as described below. Unlike in other rate-based methods we are aware of, it is now the size of the *unoccupied* memory in the receiver that determines the total of rates allocated to the senders. As in adaptive credit schemes, the current relative bandwidth usage of individual VCs determines the rates allocated to them.

We describe this rate-based control scheme in credit style, in referring to the configuration of Figure 4. Let RTT be the maximum of all the RTTs. We assume that the senders implement shaping to enforce rates of outgoing VCs as set by the receiving end of each link. A sender could be a switch, a source host, or any "virtual source" on the network edge. Below is a pseudocode for the scheme.

Notations

UM	Unoccupied Memory in the shared buffer pool at the receiver
AR	Allocated Rate for a VC
TAR	Total Allocated Rate for all VCs sharing the same buffer
MR	Measured Rate for a VC at the output. (To simplify presentation, measurement code is omitted here.)
TMR	Total Measured Rate for all VCs

Rate Setting Algorithm at Receiver

```
begin
if (UM ≤ 2*RTT*Cur_TAR) or (UM > 8*RTT*Cur_TAR)
  Update rates allocated to senders:
    New_TAR = UM/(4*RTT);
      // Note New_TAR ≤ Cur_TAR/2 or New_TAR > Cur_TAR*2,
      // respectively.
    Compute new allocated rate for each VC:
    AR = (MR/TMR)*New_TAR;
      //AR is proportional to VC's usage.
  Send updated ARs upstream
end
```

The basic idea of this credit-style rate-based scheme is that as soon as the receiver realizes that UM gets to be "too small", i.e., UM ≤ 2*RTT*TAR, a New_TAR smaller than current Cur_TAR will be computed and resulting ARs for individual VCs will be sent upstream. (One can easily see that New_TAR ≤ Cur_TAR/2.) Before the New_TAR takes effect RTT time later, some in-flight cells under Cur_TAR may still arrive. These will be no more than RTT*Cur_TAR cells. Thus, UM will never get below RTT*Cur_TAR when New_TAR takes effect. This proves that like credit control this rate-based scheme will not lose cells due to congestion.

On the other hand, when UM gets to be "too large", i.e., UM > 8*RTT*TAR, a New_TAR larger than Cur_TAR will be computed and resulting ARs for individual VCs will be sent upstream. (One can easily see that New_TAR > Cur_TAR*2.) The increased New_TAR will help improve network utilization and reduce transmission time. Note that New_TAR will not be too large that there could be a danger that it will cause buffer overflow. When New_TAR takes effect, UM will still be at least 7*RTT*Cur_TAR or 3.5*RTT*New_TAR. If in the future UM decreases and gets as low as 2*RTT*New_TAR, another smaller New_TAR will be computed and used to set new rates in time to prevent buffer overflow.

This method can achieve zero cell loss and bound the total memory requirement for achieving this. However, the method still needs careful parameter tuning to ensure fairness and efficiency under bursty traffic. For example, exactly how fast the TAR should be increased or decreased under various network and load conditions needs to be determined. Simulation results will be reported in a forthcoming report by Kung and Lin of Harvard.

8. Summary of some simulation and experimental results

Substantial simulations done by ATM Forum members have revealed performance differences between flow control mechanisms in three areas: fairness, link utilization, and switch memory use. Most of these simulations have been performed on the GFC topology shown in Figure 6, and would ideally yield the per-VC bandwidths given in Table 1. While in many circumstances credit and rate flow control perform similarly, only credit so far has demonstrated its robustness in achieving high performance under stressful conditions. Here we present a brief summary of these simulation results, with a focus on work done at Harvard. For details the reader should contact responsible researchers and look into the references.

A thorough simulation study of credit flow control [9] shows that it is almost perfectly fair not just with steady traffic in the GFC configuration, but also with highly bursty traffic over links of widely varying bandwidths and propagation delays. A study by Su, Golmie, Chang, and Benmohamed of NIST confirms the fairness and high link utilization of credit over GFC. Simulation results in [9] show that the same high performance can actually be achieved with a large number of VCs, e.g., 500 active VCs, sharing the same output link.

To evaluate the effectiveness of the adaptive buffer algorithm during severe congestion, the simulation suite in [9] includes a case where the bandwidth of a bottleneck link is suddenly reduced 100-fold. The simulation results show that during the bandwidth reduction period, not only do new VCs still ramp up quickly (modulo to link RTT), but also they ramp up in a fair manner.

Similar simulations by Bennett, Chang, Kung, and Lin of Fore Systems and Harvard, reported in the ATM Forum, pinpoint situations in which credit control has unique advantages. For instance, if one connection alternates short bursts with long silences, a competing connection can fill in all the gaps with credit control, ensuring full link utilization.

A good deal of traffic over future ATM networks may well use existing transport protocols such as TCP/IP. TCP has its own window-based flow control mechanism [7] which interprets lost or delayed packets as evidence of congestion. Studies by Fang and Chen of Sandia National Laboratories show that TCP connections over credit get close to fair shares of bandwidth and achieve full link utilization.

We have recently conducted flow control experiments on the experimental Harvard/BNR ATM switch mentioned early in Section 4. The experimental results with TCP and video confirm that credit-based flow control substantially increases efficiency under congestion. For example, for reasons similar to those discussed in Section 5, when multiple TCP sessions compete for bandwidth over a single link through a switch with a 200-cell buffer, the efficiency is only 30 percent without flow control. With flow control, the efficiency rises to over 98 percent. Flow control also helps insulate different connections, which experiments have proved to be particularly important for traffic such as video which does not provide transport-layer flow control and error correction.

Further study is needed in this area. For example, it would be of interest to understand the performance of various flow control methods for distributed computing and for large networking scenarios involving many users and a wide range of link propagation delays and bandwidths. We encourage additional simulation and field tests.

9. Fundamental issues

Credit flow control enjoys some fundamental advantages over rate-based control. We summarize them here in the hope of contributing to general understanding of flow control.

Rate-based control assumes that the rates for the circuits sharing a link can be made to converge on sensible values. However, the network load may change faster than the control system can react. On a very fast network, transfers may also take too little time to achieve a steady state. For instance, TCP sessions often transfer no more than a few dozen KBytes [13] and the required transmission time on a link of OC-3 rate, 155Mbps, is only a few milliseconds. (An extensive Unix file size survey [4] has shown that the average file length is only around 22KBytes and most files are smaller than 2Kbytes.) This situation will only get worse with increasing network capacity, and with the increasing differences in bandwidth available in different parts of the network.

For this and other reasons discussed in Section 6, rate adaptation can not and should not be precise. Since rates may not be set correctly, controlling the liability of overrunning switch buffers in an efficient and robust manner is critical to a flow control method.

In particular, when a network is heavily loaded with high-priority traffic, switches may want the option of temporarily turning off particular virtual circuits. It might seem natural to simply give a circuit a rate of zero. However, most proposed scheduling mechanisms for hosts and switches use inter-cell gaps to enforce rates. No inter-cell gap corresponds to a rate of zero. Some other mechanism may be needed to handle cases in which a switch would like to set a rate to zero. More generally, it's difficult to design hardware to provide a wide range of accurate rates.

Credit flow control is explicit about how much data a sender may transmit without receiving further credit. Lost or delayed feedback messages will not hurt, as the sender would just use the previous allowance. When necessary, the sender's transmission can be stopped completely, i.e., effectively making the rate equal to zero. When receiver-oriented adaptive buffer allocation is used, the receiver can send upstream credit information (e.g., `Fwd_Cnt`) together with the new allocation in the same management cell. If the management cell is lost or delayed, the sender would just use the previously received credit and allocation; the liability of overrunning the receiver's buffer is bounded.

Rate-based methods can be enhanced in controlling the buffer overflowing problem and thus deciding their memory requirements, by using ideas of credit control as outlined in Section 7. However, the resulting rate-based control still cannot easily handle situations in which management cells for rate updating are lost or delayed. Sources could slowly decrease the rate towards zero if no management cells arrive, but this would be at the expense of lowered network utilization and increased traffic delay. Ideally, sources should actually operate at over-allocated rates for efficiency reasons, as we have discussed in Section 6. In contrast, credit flow control does not have to worry about all these issues about how rates should be increased or decreased. Just imagine how hard it would be to make TCP work reliably and efficiently by using rate instead of window control over the Internet.

10. Conclusions

Significant differences separate credit and rate flow control. Credit provides precise control over buffer use, and can stop transmission automatically to avoid buffer overrun. Typical rate control methods provide no similar guarantee, partially to avoid some of the expense involved in implementing credit hardware. However, in the long run rate-based switches will probably need similarly complex hardware anyway, to enforce fairness and shape traffic at each switch.

Analysis, simulation and actual experiments on switching hardware show that credit flow control works well over a wide range of network conditions. At one extreme, static allocation of buffers to circuits guarantees 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. The adaptive credit 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 current predictions. In addition, credit flow control is an existence proof that congestion control can enforce guaranteed no data loss.

As our field experience with ATM networks expands, we will have much to learn, especially on 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.

Standards work and implementations should not exclude the possibility of experimenting with different flow control schemes. A system rushed to market is not likely to stand the test of time, and should have built-in possibilities for evolution. As a minimum, negotiated use of different flow-control protocols should be allowed. Effective mechanisms to let switches send out rate-management messages in time to prevent buffer overflow, such as those described in Section 7, can be studied. Moreover, a sequence number field can be included in rate-management messages. This field would be able to capture `Fwd_Cnt` values and thus allow CUP implementation. Independently, sequence numbers would be useful for the policing as discussed in Section 5.2, anyway. For example, after a switch receives a sequence number originally generated by it and later echoed back by the source, the switch can start policing.

References

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

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

[3] S. Borkar, R. Cohn, G. Cox, T. Gross, H. T. Kung, M. Lam, M. Levine, M. Wire, C. Peterson, J. Susman, J. Sutton, J. Urbanski and J. Webb, "Integrating Systolic and Memory Communication in iWarp," *Conference Proceedings of the 17th Annual International Symposium on Computer Architecture*, Seattle, Washington, June 1990, pp. 70-81.

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

[5] W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson “On the Self-Similar Nature of Ethernet Traffic,” *Proc. ACM SIGCOMM '93 Symposium on Communications Architectures, Protocols and Applications*, September 1993, pp. 183-193.

[6] M. G. H. Katevenis, “Fast Switching and Fair Control of Congested Flow in Broadband Networks,” *IEEE J. on Selected Areas in Commun.*, vol. SAC-5, no. 8, pp. 1315-1326, Oct. 1987.

[7] V. Jacobson, “Congestion Avoidance and Control,” *Proc. SIGCOMM ’88 Symposium on Communications Architectures and Protocols,* Aug. 1988.

[8] H. T. Kung, T. Blackwell and A. Chapman, “Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation, and Statistical Multiplexing,” *Proc. ACM SIGCOMM '94 Symposium on Communications Architectures, Protocols and Applications*, August 31-September 2, 1994, pp. 101-114.

[9] H. T. Kung and K. Chang, “Receiver-Oriented Adaptive Buffer Allocation in Credit-Based Flow Control for ATM Networks,” *Proc. INFOCOM ‘95*, April 1995.

[10] H. T. Kung and A. Chapman, “The FCVC (Flow-Controlled Virtual Channels) Proposal for ATM Networks,” Version 2.0, 1993. A summary appears in *Proc. 1993 International Conf. on Network Protocols*, San Francisco, California, October 19-22, 1993, pp. 116-127. (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.)

[11] C. Ozveren, R. Simcoe, and G. Varghese, “Reliable and Efficient Hop-by-Hop Flow Control,” *Proc. ACM SIGCOMM '94 Symposium on Communications Architectures, Protocols and Applications*, August 31-September 2, 1994, pp. 89-100.

[12] Larry Roberts, “Enhanced PRCA (Proportional Rate-Control Algorithm),” ATM-Forum/94-0735R1, August 1994.

[13] 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.

[14] R. J. Simcoe, “Configurations for Fairness and Other Test,” ATM_Forum/ 94-0557, 1994.