ATM_Forum/94-0282

PROJECT: ATM Forum Technical Committee
Traffic Management Subworking Group

SOURCE: H. T. Kung and Trevor Blackwell
Harvard University
& Sandia National Laboratories
Phone: (617) 496-6211
FAX: (617) 496-5508
Email: kung@das.harvard.edu

Alan Chapman
Northern Telecom
Phone: (613) 763-5221
FAX: (613) 763-2803
Email: achapman@bnr.ca

TITLE: **Adaptive Credit Allocation for Flow-Controlled VCs**

DATE: March 21-24, 1994
Raleigh Durham, NC

DISTRIBUTION: The ATM Forum Membership

ABSTRACT:

This contribution describes an adaptive credit allocation algorithm for credit-based flow control over ATM networks. At each node the allocation of credit buffers between flow-controlled VCs sharing the same switch memory can adapt dynamically to reflect their actual bandwidth usages.

There are two advantages of this adaptation capability. First, since the credit buffer size can be derived automatically, there is no need for the user or the system to specify it. This significantly eases the use and implementation of ABR services. Second, since inactive VCs can automatically yield their unused buffer space to other active ones, the total buffer size required by the flow-controlled VCs at the node can be minimized. In particular, for those VCs sharing a link their total buffer need not be larger than a small multiple of the product of the link bandwidth and round-trip link propagation delay. The total buffer can be kept the same size for a large number of flow-controlled VCs, while guaranteeing no cell loss due to congestion and ensuring high link utilization.

Simulation results demonstrating the effectiveness of this adaptive credit scheme are given.

---

1 INTRODUCTION

Flow-controlled VCs (FCVC), as described in [1, 3], promise to support ABR services effectively while maximizing network utilization. In [2] it is shown that credit-based flow control can improve the effectiveness of statistical multiplexing in minimizing switch memory.

This contribution describes an adaptive scheme for automatically deriving the proper size of the credit buffer of a VC at a node. Operating on top of basic credit-based flow control mechanism, the adaptive scheme dynamically changes the allocation of credit buffers to flow-controlled VCs sharing the same receiver memory, based on their actual bandwidth usages. Simulation results presented in this contribution demonstrate that the method can quickly adapt to load and congestion changes, and make efficient use of available memory in a switch.

2 BASIC CONCEPTS OF ADAPTIVE CREDIT ALLOCATION

In this section we give the basic concepts and intuition of adaptive credit allocation. In Section 4 we shall present an adaptive algorithm based on these ideas.

2.1 Three Types of Credit

Consider a flow-controlled VC over a link using a credit-based flow control mechanism [1, 3]. There are three kinds of credit related to the operation of the VC:

- *Allocated Credit* (*AC*). This is the size of the credit buffer allocated to the VC at the receiver. The value of *AC* defines an upper bound on the bandwidth that credit-based flow control will allow the VC to operate. This upper bound is called the “allocated bandwidth” of the VC.
- *Operating Credit (OC)*. This is the size of the credit buffer required to sustain the current bandwidth realized by the VC. This current bandwidth is called the “operating bandwidth” of the VC.
- *Used Credit (UC)*. This is the size of the credit buffer currently occupied by data cells in the receiver.

For a static or non-adaptive scheme, some fixed value of *AC* will be selected for each VC, and it is always true that $OC \leq AC$ and $UC \leq AC$. A problem with a static scheme is that it may be difficult to select the proper value of *AC* for a VC as the bandwidth of the VC may vary dynamically due to changes in offered load or in network congestion. The adaptive scheme described below alleviates this problem.

For a given link round-trip time RTT_{link}, the following relationships hold for a VC [1]:

$$\text{Allocated Credit} = \text{Allocated Bandwidth} \cdot RTT_{link}$$

$$\text{Operating Credit} = \text{Operating Bandwidth} \cdot RTT_{link}$$

Understanding these relationships, we will sometimes make interchangeable use between “allocated credit” and “allocated bandwidth”, and between “operating credit” and “operating bandwidth”. In particular, our adaptive credit allocation scheme described below will use the fact that the ratio between the operating credits of two VCs is exactly the same as the ratio between their operating bandwidths.

2.2 "Dividing a Pie"

The problem of allocating credits between the VCs sharing the same memory is like that of dividing a pie. Figure 1 depicts this analogy.

- The size of the pie corresponds to the size of the shared memory. To ensure high link utilization, we assume that the pie size is some small multiple of $RTT_{link} \cdot B_{link}$ where B_{link} is the link bandwidth. (Note that the shared memory does not need to increase with the number of the VCs in order to get high link utilization; see analysis in Section 2.5.)
- Each partition of the pie corresponds to the allocated credit for a VC (Figure 1 (a)).
- The shaded area in each partition (Figure 1 (b)) represents the operating credit of the corresponding VC. The relative ratios between the operating credits indicate the relative bandwidth usages between the VCs over some *measurement time interval* (MTI). To simplify discussion, we assume in this section that MTI is the link round-trip time given in number of cell cycles.

2.3 Adaptive Credit Allocation

Figure 1 describes how, in our adaptive scheme, credit allocation adapts to actual bandwidth usages of individual VCs. Figure 1 (a) depicts the original credit allocation between the three VCs. Figure 1 (b) shows the operating credits (denoted by shaded regions) of the VCs and their relative ratios. Note from the figure that the ratios between the operating credits are not consistent with those between the allocated credits. Figure 1 (c) shows a new credit allocation which is consistent to the relative operating credits or bandwidths of the VCs, where $\rho \leq 1$ is the ratio of the sum of the shaded areas over the pie.

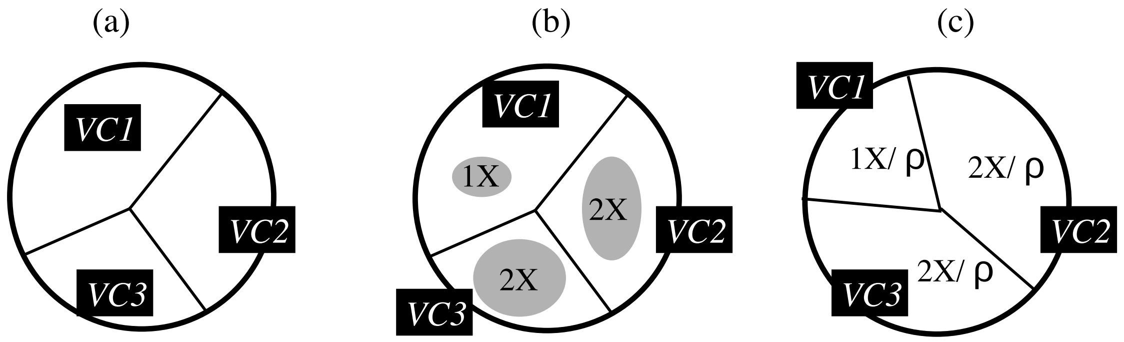

Figure 1 Adaptation of credit allocation: (a) original credit allocation between the three VCs; (b) operating credits denoted by shaded areas, and their relative ratios (1:2:2); and (c) new credit allocation consistent to the ratios of operating credits or bandwidths of the VCs.

Suppose that the new allocated credit for a VC is smaller than its current used credit. Then the new credit allocation will not take full effect until enough data cells have departed from the receiver so that the used credit is no longer larger than the new allocated credit.

2.4 Why Adaptive Credit Allocation Will Work

A key idea of the adaptive scheme described above is its use of *relative* bandwidth usages in determining new credit allocation (Figure 1 (b) and (c)). As noted in the end of Section 2.1, relative ratios between operating bandwidths of VCs are exactly the same as relative ratios between their operating credits. Relative bandwidth usages can be easily obtained by counting cell departures for individual VCs over some MTI. (Note that most likely this counting facility is already present in a switch for other purposes.)

Furthermore, the ratio-based adaptation has an important, positive side-effect described below. First notice the following facts:

- The sum of the shaded areas in Figure 1 (b) is not greater than $RTT_{link} \cdot B_{link}$, because the total operating bandwidth of all the VCs at any time cannot be larger than the link bandwidth B_{link}.
- The pie is not smaller than $RTT_{link} \cdot B_{link}$, as we assume that the size of the shared memory in the receiver is at least $RTT_{link} \cdot B_{link}$.

The above two points imply that $\rho \leq 1$ for the new credit allocation (Figure 1 (c)). The allocated credit of each VC will be larger than its operating credit by the same factor $1/\rho$ for all the VCs.

Suppose that the link is not being utilized at a high degree, i.e., ρ is substantially less than 1. Then it is desirable that a VC will ramp up its bandwidth quickly as soon as this increase becomes possible (for example, when the network becomes non congested and the VC has a burst of data to transmit). Our adaptive scheme will do precisely so, as in this situation the allocated credit of the VC will be larger than its operating credit by a large factor of $1/\rho$ The VC will have an allocated credit, which is much larger than the operating credit for the past MTI and thus have a large headroom for sustaining a much increased bandwidth in the current MTI.

Table 1 illustrates that a VC can indeed ramp up its bandwidth quickly. In this example, there are three VCs sharing the same memory in the receiver. Assume that the link bandwidth is 100 Mbps and the total allocated bandwidth for the three VCs is also 100 Mbps. (This corresponds to the case where the "pie" or the shared memory in the receiver is exactly $RTT_{link} \cdot B_{link}$.) Suppose that bandwidth VC1 and VC2 will be kept constant at 1 Mbps and 2 Mbps, respectively. Then as the table shows, the adaptive scheme allows the bandwidth of VC3 to grow rapidly from the initial 2 Mbps to 96.87 Mbps in only four MTIs.

MTI (Measurement Time Interval)	VC1		VC2		VC3	
	Operating Bandwidth	Allocated Bandwidth	Operating Bandwidth	Allocated Bandwidth	Operating Bandwidth	Allocated Bandwidth
1st MTI	1	20	2	40	2	40
2nd MTI	1	2.32	2	4.65	40	93.02
3rd MTI	1	1.04	2	2.08	93.02	96.87
4th MTI	1	1	2	2	96.87	97

Table 1: The adaptive scheme allows fast increase of VC3's operating bandwidth (in Mbps), while the operating bandwidths of VC1 and VC2 are kept constant

Figure 2 presents part of Table 1 in the form of Figure 1. Note that in Figure 2 (c) the allocated bandwidth is 93.02 Mbps because 93.02 = 100 · (40 / (1 + 2 + 40)).

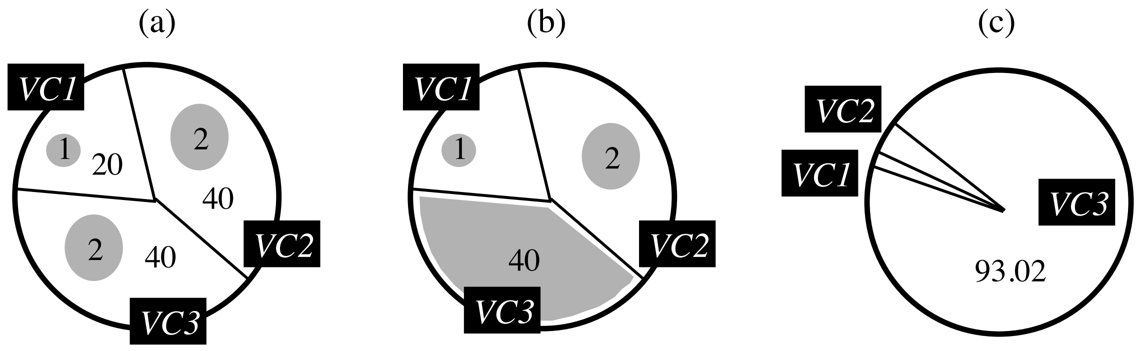

Figure 2 Adaptive credit allocation for the 1st and 2nd MTI of Table 1: (1) at the beginning of the 1st MTI allocated bandwidths of the three VCs are consistent to their operating bandwidths, (2) at the end of the 1st MTI, the operating bandwidth of the VC3 can increase to 40 Mbps, and (3) at the beginning of the 2nd MTI, the allocated bandwidth can thus increase to 93.02 Mbps

Suppose that the total allocated bandwidth for the three VCs is 200 Mbps instead of 100 Mbps, that is, the receiver's memory is twice as large as $RTT_{link} \cdot B_{link}$. Then it is easy to check that VC3 can ramp up at an increased speed, i.e., VC3 can reach 97 Mbps in just three MTIs. In general, we assume that the "pie" is strictly larger than $RTT_{link} \cdot B_{link}$, so that good ramp up speeds can be expected for any VC.

2.5 A Simple Analysis

We give an analysis for the fast ramp-up result illustrated in Table 1. The following notations are used:

- X = Current operating bandwidth of the VC which is ramping up. (This VC is VC3 in Table 1.)
- C = Total bandwidth of all the other VCs whose bandwidths will be kept constant. (These VCs are VC1 and VC2 in Table 1.)

Suppose that the receiver's memory is exactly the same as $RTT_{link} \cdot B_{link}$. Then the allocated bandwidth of the ramping up VC will be $B_{link} \cdot X / (C + X)$ at the end of the current MTI. This implies by the adaptive algorithm that the operating bandwidth of the VC will be $B_{link} \cdot X / (C + X)$ for the next MTI. Thus, after the current MTI, the bandwidth of the VC will be ramped up by a factor of:

$$[B_{link} \cdot X / (C + X)] / X = B_{link} / (C + X) \qquad (1)$$

We are interested in how fast X can ramp up from a small value to a large value. For this purpose, it is without loss of generality to assume that B_{link} is a large and C is relatively small (e.g., in Table 1, $B_{link} = 100$ and $C = 3$). Note that when X is small, Equation (1) implies that the ramping up factor for each new MTI is large. As X increases, the ramping up factor decreases. However, when X is so large that the ramping up factor is near 1, it is also the time when the operating bandwidth of the VC, $B_{link} \cdot X / (C + X)$, is already close to its maximum-possible value, B_{link} - C.

Suppose that the receiver's memory is $k \cdot RTT_{link} \cdot B_{link}$, for some constant $k > 1$. Then instead of Equation (1), the ramping up factor will be $k \cdot B_{link} / (C + X)$. This explains why the ramping up speed increases as the memory size, as noted in the end of Section 2.4.

2.6 Use of Statistical Multiplexing in Minimizing Switch Memory

Adaptive credit allocation can use statistical multiplexing for minimizing switch memory in a natural way. This is accomplished by simply letting the "pie" from which all VCs' credits are allocated be *larger* than the *real* memory at the receiver. By using a large "pie", the adaptive scheme will be able to ramp up individual VCs at an increased speed (as shown in Section 2.5), and to accommodate an expanded number of VCs. However, there will be some risk of memory overflow at the receiver and thus cell loss.

Note that credit-based flow control can improve the effectiveness of statistical multiplexing in minimizing memory usage. This is basically because credit-base flow control automatically limits burst sizes in carried traffic to the credit size. Simulations in [2] have shown that with credit-based flow control, cell loss can be negligible even when the "pie" is an order of magnitude larger than the receiver's real memory.

We summarize our general approach of improving the utilization of switch memory or minimizing its size as follows. When a number of VCs share a receiver's (real) memory, the memory can become under utilized for two reasons: (1) memory is allocated to inactive VCs which are not transmitting; and (2) memory is allocated to those active VCs which are not blocked by congestion and as a result do not actually occupy the allocated memory. Whereas adaptive credit allocation will assure that problem (1) will not become significant, using a pie larger than receiver's real memory is a step towards solving problem (2) by statistical multiplexing.

3 A BRIEF REVIEW

To give a complete presentation in this contribution, we briefly review some related background information in this section. After this background information is presented, we will be ready in Section 4 to describe an adaptive credit allocation scheme implementing the ideas of Section 2.

3.1 Basic FCVC Scheme

The basic FCVC (Flow-Controlled VC) scheme [1] is a credit-based flow control method. It generally works over a flow-controlled VC link as follows. Before forwarding any data cell over the link, the sender needs to receive credits for the VC via credit cells sent by the receiver. At various times, the receiver sends credit cells 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.

We briefly review here a particular credit-based flow control method which is called the *N23* Scheme in [1]. The receiver is eligible to send a credit cell (to the sender) for a VC each time after it has forwarded *N2* data cells of the VC since the previous credit cell for the same VC was sent. The *N2* value is a design or engineering choice. For example, a reasonable value for *N2* could be 10, so that a link would never consume more than 10% of its bandwidth in transmitting credit cells.

The credit cell contains "credit information" reflecting available buffer space at the receiver for the VC. Upon receiving the credit information, the sender updates its new credit balance for the VC. In the *N23* Scheme, the new credit balance will be the number of additional data cells that the sender can forward without the risk of overflowing the buffer corresponding to the VC at the receiver.

The buffer for the VC at the receiver is assumed to have $N2 + N3$ cells. The *N3* value is chosen to be just large enough to avoid data and credit underflow, so that the flow control mechanism itself will not prevent the VC from sustaining its peak targeted bandwidth B_{vc}. That is,

$$N3 = RTT_{link} \cdot B_{vc} / Cell\_Size$$

Note that in the terminology of Section 2, the allocated credit for the VC would be $N2 + N3$. Increasing the value of *N3* will increase the bandwidth of the VC that flow control will allow, but will also increase the memory allocation required by the VC.

3.2 Static CUP

We describe here a static CUP (Credit Update Protocol)—a variant of that presented in [2]—on which the adaptive credit allocation scheme of the next section will be implemented. The static CUP is an implementation of the basic *N23* scheme described in Section 3.1.

Consider per VC flow control over a link. The sender or receiver keeps a running total V_s or V_r, respectively, of all the data cells it has forwarded for each flow-controlled VC. The receiver will enclose the up-to-date value of V_r in each transmitted credit cell for the VC. When the sender receives the credit cell with value V_r, it will update the Credit_Balance for the VC by:

$$\text{Credit\_Balance} = N2 + N3 - (V_s - V_r) \qquad (2)$$

Note the explicit dependence of the Credit_Balance on $N2 + N3$. This is convenient for accommodating adaptive adjustment of *N3* described in the next section.

4 ADAPTIVE CUP

We now describe an adaptive CUP (Credit Update Protocol) implementing the basic adaptation concepts of Section 2. The adaptive scheme will automatically adjust the *N3* value (of Equation (2)) for a VC to reflect its actual bandwidth usage at a given time. Using this scheme a VC will automatically decrease its *N3* value, if the VC does not have sufficient data to forward or is back-pressured because of downstream congestion. The freed up buffer space will automatically be assigned to other VCs which have data to forward and are not congested.

The *N3* adjustment algorithm is run periodically by software. The algorithm cycles through all VCs involved in the adaptive credit protocol: for every iteration of the algorithm, the *N3* of a single VC is updated. The number *N3T*, which stands for "*N3* Total", represents the size of the memory that these VCs share in the receiver less $N2 \cdot N$ with N being the total number of these VCs. ($N3T + N2 \cdot N$ corresponds to the "pie" size of Section 2.) The value of *N3T* may be statically configured, or dynamically varied by some other protocol using relatively large time constants.

Suppose that $N3T + N2 \cdot N \leq M$, where M is the size of the real memory in the receiver. Then it can be guaranteed that there will be no cell loss due to congestion. However, as noted in Section 2.6, one may choose to let $N3T + N2 \cdot N$ be greater than M and accept some low rate of cell loss, in order to take advantage of statistical multiplexing in minimizing the memory.

The adaptive algorithm computes a target *N3* for a VC as a fraction of the *N3T*, proportional to the VC's fraction of egress traffic on the link. The value is limited by a minimum and maximum value to prevent *N3* going to zero for inactive VCs, and to prevent any VC from exceeding a limit. The algorithm ensures that two properties always hold:

- The total *N3* for all VCs is no greater than *N3T*.
- The number of cells of a given VC actually present in the receiver's memory is no more than the *N2* + *N3* for that VC.

The second property is ensured by never decreasing a VC's *N3* such that its current credit amount becomes negative. Thus, the memory use in the receiver is bounded at $N3T + N2 \cdot N$. In fact, it rarely seems to exceed 1/3 of *N3T*, because credit-based flow control will in general improve the effectiveness of statistical multiplexing in minimizing memory usage [2].

Both the aggregate link bandwidth, and operating bandwidth of individual VCs are measured by counting the number of cells over a period MTI, typically a few round-trip link times.

The parameter α defines a single pole low-pass IIR filter, which insulates *N3* values from noise in the egress traffic. In our simulations we set α close to 1, but lower values may improve performance for traffic sources expected to be fairly smooth.

Adaptive *N3* Algorithm

```
vcID := a VC to adjust

bandwidthFraction := # cells sent for the given VC, divided by the total number of cells
        sent over the last MTI cell times

targetN3 := $(1- \alpha[\text{vcID}]) \cdot \text{N3}[\text{vcID}] + \alpha[\text{vcID}] \cdot \text{bandwidthFraction} \cdot \text{N3T}$

limit targetN3 to be:
        no less than minN3
        no more than maxN3

targetDelta := targetN3 - N3[vcID]

limit targetDelta to be:
        no less than (creditAmount[vcID])
        no more than (totalN3 - N3Sum)

increase N3[vcID] by targetDelta
```

```
        increase N3Sum by targetDelta
        increase creditAmount[vcID] by targetDelta
```

We note some properties of the adaptive scheme that make it easy to implement. It requires no communication beyond the credit messages specified for the CUP scheme in Section 3.2. The sending node only needs to know how much memory it is allowed to use in the receiving node. We expect that much can be done to tune this algorithm for optimum performance in various networking configurations.

5 SIMULATION RESULTS: ADAPTIVE CREDIT ALLOCATION BASED ON BANDWIDTH USAGES

This section reports simulation results demonstrating the effectiveness of the adaptive *N3* algorithm described in Section 4.

5.1 Simulation Configuration (for S1 and S2)

The basic simulation configuration (for Simulation S1 and S2) below is shown in Figure 3. This configuration was carefully chosen, so it is sufficiently simple to allow easy interpretation of simulation results and also it is sufficiently general to cover most of the key issues we want to address.

There are *N* VCs originating from some number of source hosts (indicated by shaded rectangles) and passing through two switches (indicated by shaded circles). Multiple VCs may enter Switch-1 at the same input port. All the VCs depart from the same output port of Switch-1 so there will be congestion at the port which we study. (In Figure 3, each solid bar indicates a switch input or output port.) All the VCs will share the same *M*-cell memory in Switch-1. (Switch-1 is referred to as "the switch" in the rest of the paper, unless otherwise is stated explicitly.) We assume a simple output buffered switch where VCs can be individually scheduled and accessed at each output port.

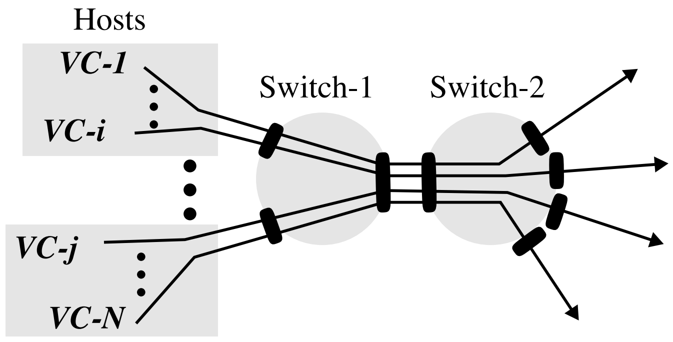

Figure 3 Simulation configuration

5.2 Notations

B: VC burst size (# cells)
D: Round-trip propagation delay between each host and the switch (# cells)
F: Total offered load on the average (% of the link bandwidth of the switch output port).
M: Size of switch memory (# cells)
MMU: Maximum memory usage (# cells) for a given traffic load
N: Number of VCs
T: Simulated time (# cell cycles)

5.3 Load Assumptions

The *N* VCs have identical load involving *B*-cell bursts. The average offered load of each VC is therefore 1/*N* of the total offered load.

5.4 Simulation S1

B (VC burst size) = 172 (approximately an 8K-byte file transfer block)
D (Round-trip propagation delay) = 3,200
F (Total offered load) = 95%
M (Switch memory size) = Unlimited
N (Number of VCs) = 100
T (Simulated time) = 1,000,000

Figure 4 shows the memory usage in Switch-1 as a function of time in cell cycles. Memory usage is much lower than 3,200, the round-trip link delay. (Similar results are shown in Table 2.) The low memory usage is achieved without any loss of cells due to congestion.

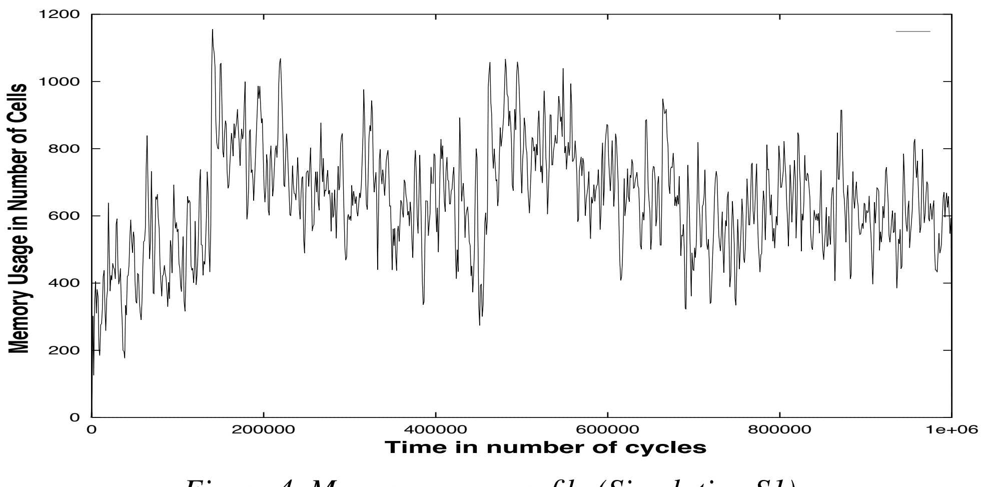

Figure 4 Memory usage profile (Simulation S1)

Figure 5 plots the *N3* values of a VC against iterations of the adaptive credit algorithm—one iteration is 500 cell cycles. (Note that the allocated credit for the VC is *N2* + *N3* = 10 + *N3*, as in our simulations *N2* is set to be 10.) The *N3* values change rapidly to adapt to load changes. Note that changes of the *N3* values at Switch-1 closely track those at the source host, and those at Switch-2 closely track those at Switch-1. (The three traces are so close that they almost coincide in Figure 5.) A blown up version of part of this tracking is shown in Figure 6.

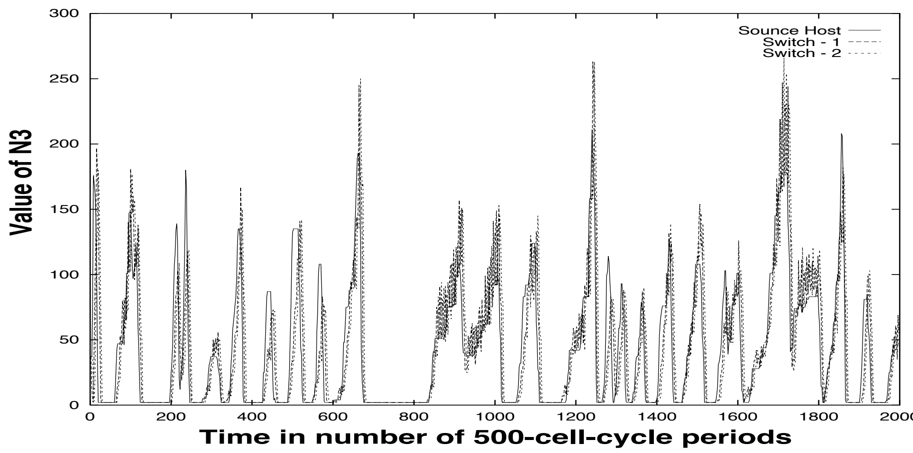

Figure 5 Changes of the N3 value at the source host, Switch-1 and Switch-2 track each other closely (Simulation S1).

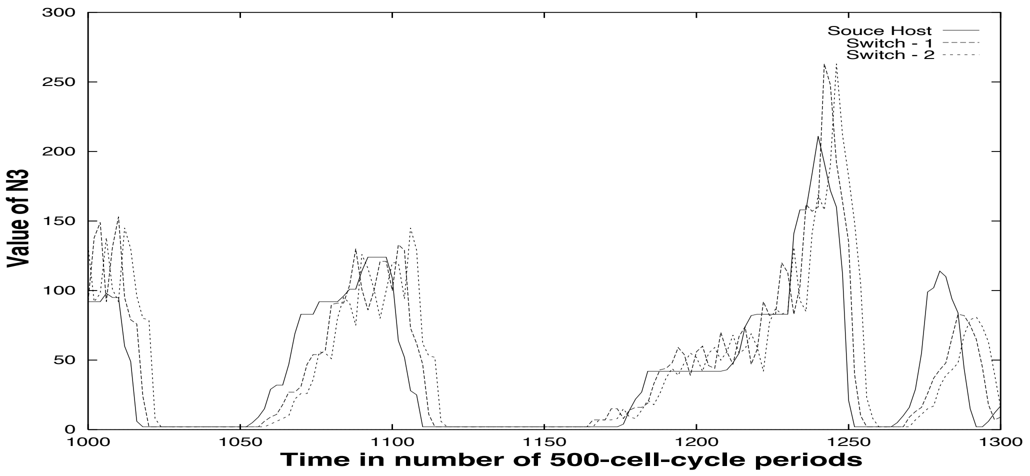

Figure 6 A blown up view of part of the tracking shown in Figure 5 (Simulation S1)

Notice that the *N3* values of consecutive switches track each other with only a short delay. As can be seen in Figure 6, a switch lags behind the switch upstream by no more than 5,000 cell cycles. This reaction time is only a few round trip times (recall that a round-trip time is 3,200 in the simulation). Thus the increase of the *N3* value in response to an arriving burst of cells is indeed rapid as predicted by the analysis of Section 2.

The peak *N3* reached in response to a burst is about equal to the burst size (172) in most cases. Figure 5 shows that it never exceeds twice the burst size. The adaptive *N3* can reach values much higher than would be practical for a statically defined *N3*. This allows the switch to clear the burst through as quickly as possible, reducing memory usage.

5.5 Simulation S2

B (VC burst size) = 172
D (Round-trip Propagation delay) = 3,200
F (Total offered load) = 95%
M (Switch memory size) = Unlimited
N (Number of VCs) = 200
T (Simulated time) = 1,000,000

	MMU	# Cell Delays	Link Utilization
Adaptive Credit Allocation	1,150	14,088	95%
Static Credit Allocation	1,600 (*N3*=30) 2,300 (*N3*=40)	16,083 (*N3*=30) 13,272 (*N3*=40)	95%

Table 2: Performance comparisons between adaptive and static credit allocation (Simulation S2)

Table 2 summarizes the performance of the adaptive credit allocation method, in terms of its *MMU,* delay and link utilization. Again, note that relatively small memory usage is achieved.

5.6 Simulation S3

This simulation is for NFS traffic depicted in Figure 7. Clients on the right-hand side issues read requests to the servers on the left-hand side. The figure shows 80 VCs each carrying data responding to requests carried on paired VCs from right to left; the traffic on client-to-server VCs is small. NFS traffic is essentially a request/response transaction stream, which is typical of many database access and transaction processing protocols. In this simulation, we measured the total number of transactions completed between clients and servers, as well as the number of datagrams lost.

Data Packet Size = 172 · 4 = 728 (approximately, 32K-byte block)
D (Round-trip propagation delay) = 3,200
N (Number of VCs) = 80
T (Simulated time) = 1,000,000
N2 = 10
N3 = Adaptive

Without flow control, a very large amount of memory is required to achieve acceptable datagram loss levels. Flow control reduces the loss levels to zero while using substantially less memory. In the simulation, the population of users is large enough that the lost traffic of a few

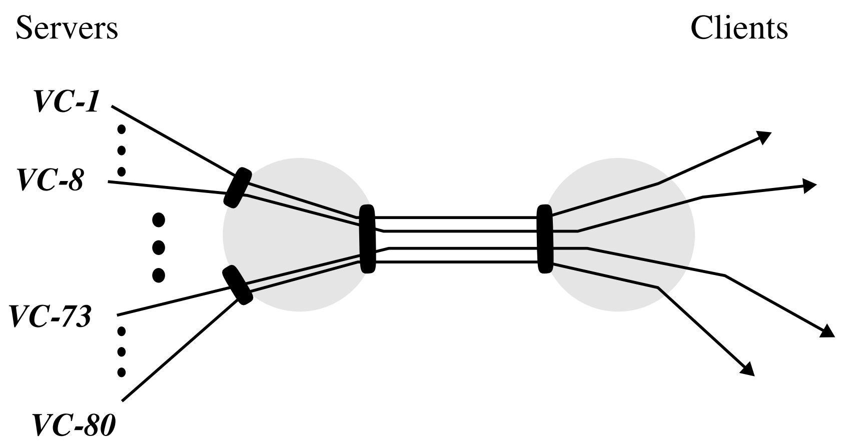

Figure 7 NFS simulation configuration (Simulation S3)

	FC Adaptive *N3*	Non FC
Memory Size (*M*)	Cell Loss Rate & Datagram Loss Rate (L & DL)	
3,700 Cells 10,000 Cells	0% & 0% 0% & 0%	6% & 4.5% 4.1% & 2.6%
	# Complete Transactions	
5,000 Cells	2,681	1,907
	Datagram Transit Time (Cell Times)	
	16,700	7,300

Table 3: Performance comparison between FC and non FC implementations of NFS (Simulation S3)

clients does not impact the total transaction rate substantially. In real NFS, every lost datagram causes a pause of 1 to 5 seconds for some client. These pauses can be quite annoying to users.

Assuming OC-12 links and a 3,700-cell switch memory, the network latency is 22 ms for FC, and 10 ms for non FC. With FC, if a client process needs to make 1,000 serial reads, total time will be 22 seconds in addition to server time. Without FC, assuming losses cause a 2 second pause, total time will be 100 seconds.

6 WHAT DO ALL THE ABOVE SIMULATION RESULTS MEAN?

We draw some general insights from the simulations reported in the preceding section and from other simulations which we have done but are not reported in this contribution. Table 4 summarizes these conclusions.

	Non FC	Adaptively FC	Statically FC
Delay	*x*	*2x*	*2x*
Memory Usage	>> *y*	*.5y*	*y*
Ease of Use	High	High	Low

Table 4: Performance comparison between non flow-controlled (Non FC), adaptively flow-controlled (Adaptively FC), and statically flow-controlled (Statically FC) networks

Consider an offered traffic giving say, $F = 95\%$ load, such as that used by a S1 or S2 simulation in the preceding section. Suppose that a non flow-controlled network (with unlimited memory and per-VC queueing) has an average delay of *x*. Then if the network is adaptively flow-controlled (by the *N3* adaptive algorithm of Section 4), the average delay is expected to be no more than *2x*. Moreover, for bursty traffic this adaptively flow-controlled network is expected to need only about a half of the memory (having, say, *y* cells) that is required by the corresponding statically flow-controlled network achieving the same average delay. The required memory for the non flow-controlled network to achieve a reasonably low cell loss rate will need to have much, much more than *y* cells. Both the non flow-controlled and adaptively flow-controlled networks are easier to use than a statically flow-controlled network, because they don't require users' assistance in allocating credit buffer (i.e., setting the *N3* value). From this analysis, we conclude that the adaptive FC is the winner among the three approaches.

7 CONCLUDING REMARKS

The adaptive credit allocation scheme described in this contribution works. It offers a strong indication that ATM networks can effectively provide *true* ABR services which require absolutely no *a priori* contracts between the network and end users. In particular, the adaptive scheme achieves the following objectives:

- Hosts need not estimate how much bandwidth they would subscribe for an ABR service in order to request an appropriate credit buffer size.
- Idle or relatively inactive VCs will not consume any significant switch memory.
- Traffic requiring large peak bandwidths but with low average bandwidth (X Window connections, for example) can use network resources efficiently.

The adaptive *N3* protocol used in the simulations of this contribution is expected to be improved and extended, although it is already effective as shown by the simulation results. As in studying any adaptive schemes, this is an intellectually rich area deserving deep investigation.

The general adaptive framework presented in this contribution involves only some simple adaptation principles and bandwidth measurements. The framework has the built-in flexibility for further tuning and optimization.

REFERENCES

[1] Kung, H. T. and A. Chapman “The FCVC (Flow Controlled Virtual Channels) Proposal for ATM Networks”. A summary appears in *Proc. 1993 International Conf. on Network Protocols*, San Francisco, California, October 19-22, 1993, pp. 116-127. Postscript file of the paper is available via anonymous FTP from virtual.harvard.edu:/pub/htk/atm-forum/fcvc.ps.

[2] Chapman, A, Kung, H. T. and T. Blackwell, “Use of Flow Control for Effective Statistical Multiplexing and Notes on Implementation,” ATM Forum Contribution 94-0085, ATM Forum Meeting, Lake Tahoe, Nevada, January 17-21, 1994. Postscript file of the paper is available via anonymous FTP from virtual.harvard.edu:/pub/htk/atm-forum/atm_forum_94_0085.ps.

[3] Jim Scott and Bob Simcoe, “Digital Flow Control,” ATM Forum Contribution 93-778.