The FCVC (Flow-Controlled Virtual Channels) Proposal for ATM Networks: A Summary

H. T. Kung
Division of Applied Sciences, Harvard University, 29 Oxford Street, Cambridge, MA 02138

Alan Chapman
Bell-Northern Research, P.O.Box 3511, Station C, Ottawa, Ontario K1Y 4H7

Abstract

Link-by-link flow-controlled virtual channels are proposed for asynchronous transfer mode (ATM) networks. The flow control mechanism efficiently uses leftover bandwidth after guaranteed traffic has been served, thus achieving high network utilization. Compatible with existing ATM standards, this proposal is called FCVC, which stands for Flow-Controlled Virtual Channels.

Three increasingly memory-efficient credit-based flow control schemes, named N123, N123+ and N23, are described for implementing link-by-link flow control. These credit-based flow control schemes have been used in an experimental 622-Mbps ATM switch currently under joint development by BNR and Harvard. Simulations based on the switch design have confirmed that the flow control mechanism is indeed able to provide sufficiently rapid feedback to let a network adapt to load changes and maximize its performance. This paper is a summary of [13].

1. Introduction

A fundamental reason for per virtual circuit (VC), link-by-link flow control (LLFC) is to provide effective means of using filling-in traffic to maximize network utilization, as depicted in Figure 1. Typically, best-effort traffic is used to fill in bandwidth slack left by scheduled traffic with guaranteed bandwidth and latency such as video and audio.

To allow effective traffic fill in, fast congestion feedback for individual VCs is needed. Measurements have shown that data [8, 15] and video [9] traffic often exhibit large bandwidth variations even over time intervals as small as 10 milliseconds. With the emergence of very high-bandwidth traffic sources such as high-speed host computers with 800-Mbps HIPPI network [4] interfaces,

This research was supported in part by BNR, 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.

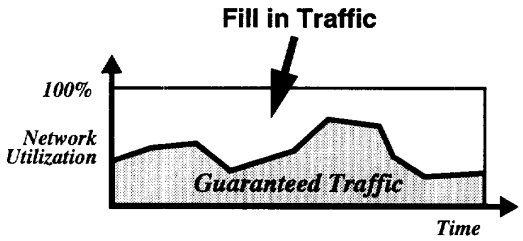

Figure 1: Flow control for effective fill in of traffic in bandwidth slacks

networks will experience further increases in load fluctuations [12]. To utilize slack bandwidth in the presence of highly bursty traffic, fast congestion feedback is necessary. LLFC implements the required feedback at the fastest possible speed.

Another reason for LLFC is congestion control. For high-speed networks there is the problem of increased mismatches in bandwidth [12]. When the peak speed of links increases in a network, so may bandwidth mismatches in the network. For example, when a 1-Gbps link is added to a network which includes a 10-Mbps Ethernet, there will be two orders of magnitude difference in their speeds. When data flows from the high-speed link to the low-speed one, congestion will build up quickly. This represents additional congestion scenarios beyond the usual congestion caused by the merging of multiple traffic streams.

The highly bursty traffic and increased bandwidth mismatches expected will increase the chance of transient congestion. It is therefore important to ensure that transient congestion does not persist and evolve into permanent network collapse.

Using LLFC, a VC can be guaranteed not to lose cells due to congestion. When experiencing congestion, backpressure will build up quickly along congested VCs spanning one or more hops. When encountering backpressure, the traffic source of a congested VC can be throttled. Thus excessive traffic can be blocked at the boundary of the net-

0-8186-3670-X/93 $03.00 © 1993 IEEE

work, instead of being allowed to enter the network and cause congestion problems to other traffic.

The "per VC" LLFC allows multiple VCs over the same physical link to operate at different speeds, depending on their individual congestion status. In particular, congested VCs cannot block other VCs which are not congested.

The throttling feature on individual VCs, enabled by LLFC, is especially useful for implementing high-performance, reliable multicast VCs. At any multicasting point involving more than a moderate number of ports, the probability that one or more of them are not available at a given cell cycle is likely to be high, and the delay before a cell is forwarded to all the ports can fluctuate greatly. It is therefore essential for reliable multicast VCs to throttle in order to accommodate the inherent high variations in their transmission speeds [13].

In addition, flow control will allow new services for hosts with high-speed network access links operating, for example, at 100 megabits per second. For instance, these hosts can be offered a new kind of data communications service, which may be called a "greedy" service, where the network will accept as much traffic as possible at any instant from VCs under this service. Flow control can be used to throttle these VCs on a per VC basis when the network load becomes too high, and also to speed them up when the load clears. There will be no requirements for predefined service contract parameters, which are difficult to set dynamically. This "greedy service" is expected to serve many types of best-effort traffic effectively and efficiently.

2. Summary of Results and Organization of the Paper

Three credit-based flow control schemes (called *N123*, *N123+* and *N23*) for implementing link-by-link flow control are proposed in this paper, for asynchronous transfer mode (ATM) networks [1, 3]. These are three increasingly memory-efficient schemes with increasing implementation cost.

Defined in terms of ATM cells, the proposed credit-based flow control schemes can operate on top of various underlying physical media and over non flow-controlled networks by using "tunneling" [13]. The schemes can efficiently implement flow-controlled VCs of any bandwidth, and can also limit the bandwidth overhead to support flow control to be less than any given fraction of the total link bandwidth.

Moreover, the schemes are robust in the sense that they can recover automatically from link errors. Adjustments can be easily made to increase the degree of automatic protection against errors at the expense of increased bandwidth overhead or buffer memory size.

With the proposed credit-based schemes, a local area network node can support on the order of 1,000 flow-controlled VCs each capable of peaking at a rate of 100 Mbps, using only about one megabyte of buffer memory in total. The credit-based schemes also apply to networks with large propagation delays, using proportionally larger memories.

The proposed credit-based flow control schemes have been implemented in an experimental ATM switch design currently under joint development by BNR and Harvard. Performance simulations carried out on a simulator reflecting the hardware design have confirmed the effectiveness of the proposed schemes [14].

The rest of the paper is organized as follows. First an overview of a credit-based flow control approach is given. Then the *N123*, *N123+* and *N23* Schemes are described. The *N23* Scheme is the most significant and interesting among the three methods, as it uses the smallest amount of memory, and is equivalent to the "additive" credit updating method [5, 6, 11] as far as the effect of flow control on buffer management is concerned.

However, the *N123* and *N123+* schemes can also be attractive due to their relatively simple implementation. Their extra buffer space is proportional to the link propagation delay, and can be of no practical concern for local area networks as they have small propagation delays.

This paper will give a detailed description of the *N23* Scheme and prove some of its mathematical properties. But due to space limitation, the paper will only briefly describe the *N123* and *N123+* Schemes, for which details can be found in [13].

3. Credit-Based, Per VC Link-by-Link Flow Control

Flow control based on credits is an efficient way of implementing per VC LLFC. A flow-controlled VC is composed of one or more flow-controlled *VC links* or simply *links*, connecting various network subsystems such as switches and adapters. Figure 2 depicts two flow-controlled VCs (*VC1* and *VC2*) for which credit-based flow control is used for each link.

Figure 3 is a magnified view of the two flow-controlled VC links between Adapter 1 and Switch 1. During the operation of a VC, two types of ATM cells, called *data* and *credit cells*, will be used. A data cell transports data belonging to the VC. A credit cell transports credit values and various credit-related management information for the VC. All credit cells are transported over some reserved VCs, called the *credit-carrier VCs*. Refer to [13] for a pro-

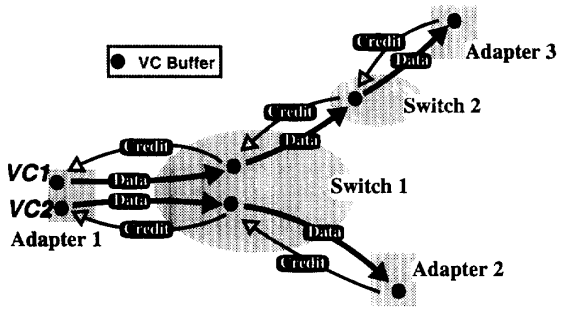

Figure 2: Credit-based flow control applied to each link of a VC

posed credit cell format, compatible with AAL type 5 [1], and some credit-related transaction types.

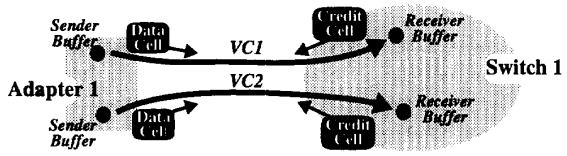

Figure 3: Magnified view of two flow-controlled VC links in Figure 2

Each VC link is associated with a pair of *sender* and *receiver buffers*, which are also called *VC buffers*. (VC buffers are denoted by dots in Figures 2 and 3.) Transporting data cells from the sender buffer to the receiver buffer over the VC can be *flow-controlled* to prevent overrun of the receiver buffer. For two consecutive links of the same VC, the receiver buffer of the upstream link is also used as the sender buffer of the downstream link.

The credit-based flow control over a VC link generally works 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 that there is a certain amount of buffer space available for receiving data cells of the VC. After having received a credit value, the sender is eligible to forward some number of data cells of the VC to the receiver according to the received credit value. Each time the sender forwards a data cell of a VC, it decrements its current credit count for the VC by one.

When receiving a credit cell for a VC the sender updates its credit count for the VC using an *absolute* updating method. That is, the new credit count will be computed entirely from the newly received credit, independently of the old credit count. In particular, the new credit count will not be relative to the old credit count. This is in contrast with relative, incremental or additive updating used in some previously proposed "credit-like" flow control schemes [5, 6, 11], where the new credit count is equal to the old credit count plus the newly received credit value. The absolute credit updating allows a robust flow control scheme in the sense that any effect of a corrupted credit will be repaired automatically by the arrival of the next successfully transmitted credit (see Section 8.8). The three credit-based flow control schemes (*N123*, *N123+* and *N23*) described in this paper all use absolute credit updating.

4. Definitions and Terminologies

For describing the credit-based flow control schemes of this paper, it is convenient to consider three consecutive nodes of a VC referred to as *upstream*, *current* and *downstream* nodes. Figure 4 depicts these nodes along with their VC buffers. With respect to the link between the upstream and current nodes, they are the sender and receiver nodes, respectively. Similarly, with respect to the link between the current and downstream nodes, they are the sender and receiver nodes, respectively.

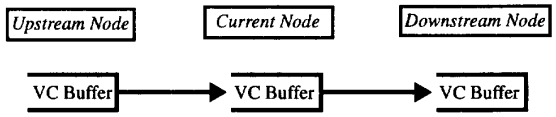

Figure 4: Three consecutive nodes of a VC and their VC buffers

The following definitions and notations will be used throughout:

- R = Round-trip link delay (in seconds) between the current and upstream nodes, including both the link propagation delay and the time for handling data cells and processing credit cells at the two endpoints. (The time at the two endpoints can be a constant and can be as small as several microseconds as in the case of the BNR/Harvard switch.) R is a system parameter, which can be determined at the system configuration time, and can be measured by executing a built-in looping routine. Overestimating R causes no real harm in the sense it may introduce occasional data and credit underflow (see Section 5), but will never introduce buffer overflow [13]. In practice, it should be easy to overestimate R within one cell cycle time as in the BNR/Harvard switch.
- F = Number of in-flight data cells not accounted for by the credit cell in question.
- B_{VC} = Targeted bandwidth (in bits per second) of a VC over time R.
- B_{link} = Peak bandwidth (in bits per second) of the underlying physical link over time R.
- *Cell_Size* = 424, which is the number of bits in an 53-byte ATM cell.

Note that $B_{VC} \leq B_{link}$ is always true, and in general, B_{VC} can be much smaller than B_{link}. This is the reason why the *N123+* and *N23* Schemes of Sections 7 and 8 are designed so that the VC buffer size for a VC can be bounded above by a quantity proportional to B_{VC} rather than B_{link}.

5. The *N1*, *N2*, and *N3* Zones of VC Buffer

For each VC crossing the current node, its VC buffer can be viewed to consist of three zones. As depicted in Figure 5, they are called *N1*, *N2*, and *N3* zones, each possessing *N1*, *N2*, and *N3* cells, respectively.

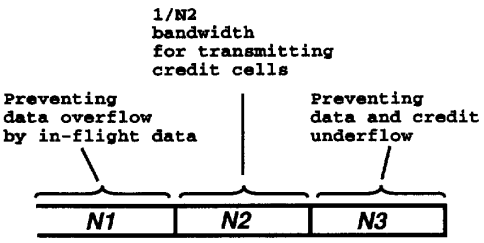

Figure 5: Three zones of each VC buffer

- The *N1* zone covers the round-trip link delay between the current node and the upstream node so that data overflow of the VC buffer can be prevented. That is, the *N1* zone will catch all in-flight data cells of the VC which the upstream node could have forwarded before it receives the most recent credit cell for the VC from the current node. It turns out that from [13] and the next three sections of this paper:
  - For the *N123* Scheme:

    $N1 = R \cdot B_{link} / Cell\_Size$

  - For the *N123+* Scheme:

    $N1 = min\ (N2 + R \cdot B_{VC} / Cell\_Size,\ R \cdot B_{link} / Cell\_Size)$

  - For the *N23* Scheme:

    $N1 = 0$

- The *N2* zone allows less frequent sending of credit cells (while still preventing data and credit underflow) to reduce the bandwidth overhead for transmitting credit cells. More precisely, the current node is eligible to send a credit cell for a VC to the upstream node, only after the current node has forwarded *N2* data cells of the VC to the downstream node, since the last credit cell for the VC was sent. Note that the frequency of sending credit cells of the VC depends on the value of *N2* and on the speed at which the current node forwards the data cells of the VC.
  - The value of *N2* can be the same for all flow-controlled VCs in a network. The value can be a design or engineering choice. For example, a reasonable value for *N2* could be 10, so that a link will never consume more than 10% of its bandwidth in transmitting credit cells.
  - It is also possible that different VCs use different *N2* values. For example, a VC for which automatic recovery from errors is critical may choose to use a relatively small *N2* in order to increase the chance that another credit cell will immediately follow a corrupted credit cell. See Section 8.8 for further discussions on the error recovery issue.
- The *N3* zone prevents data and credit underflow, so that the VC can sustain its targeted bandwidth as long as the upstream node has data to forward and the downstream node has space to receive them.
  - The *N3* zone must be large enough to prevent data underflow. It must hold enough data cells to continue sending downstream at the targeted bandwidth, B_{VC}, while waiting for new data cells from upstream reflecting the most recently sent credit cell on this VC.
  - To prevent credit underflow, the upstream node must receive credit cells each with sufficiently large credit value. This prevents the upstream node (when operating at the targeted VC bandwidth) from depleting its current credit before the next credit cell arrives. Note that as described in Section 8, each new credit value is *N2* + *N3* less the buffer fill. Thus, for a given value of *N2*, the *N3* value must be large enough to allow use of sufficiently large credit values.

  It turns out from [13] and the next three sections of this paper that for *N123*, *N123+* and *N23* Schemes, to prevent both data and credit underflow it suffices to choose the value of *N3* to be:

  $$N3 = R \cdot B_{VC} / Cell\_Size \quad (1)$$

6. The *N123* Scheme: Basic Credit-Based Flow Control Scheme

The "*N123* Scheme" is the first of the three credit-based flow control schemes of this paper. These methods have a number of desirable features such as provision for transmitting credit cells at any low bandwidth and robustness against corrupted credit cells. In addition, these methods have the usual features such as prevention of buffer overflow and underflow typically found in other flow control methods.

The *N123* Scheme represents a baseline, credit-based flow control method, which is relatively easy to understand. The other two schemes, *N123+* and *N23*, of Sec-

tions 7 and 8 provide additional beneficial features at the expense of some added implementation cost.

Figure 6 depicts the basic algorithm for the *N123* Scheme.[1] The current node is eligible to send a credit cell (to the upstream node) for a VC each time after it has forwarded at least *N2* data cells of the VC (to the downstream node) since the previous credit cell for the same VC was sent. The credit cell will contain a credit value (for the VC) equal to the number of unoccupied cell slots in the combined area consisting of the *N2* and *N3* zones[2]. A credit cell need not be sent when the combined area is totally occupied.

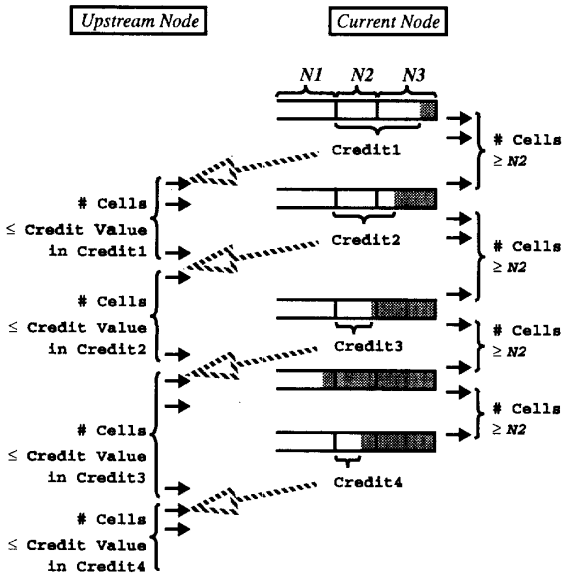

Figure 6: *N123* Scheme

Upon receiving a credit cell with credit value *C* for a VC, the upstream node is permitted to forward up to *C* data cells of the VC before the next successfully transmitted credit cell for the VC is received. Specifically, the upstream maintains a count, called Credit_Count, for the VC. Credit_Count could be set to be *N2* + *N3* initially. Each time the upstream node forwards a data cell of the VC (to the current node), it decrements the Credit_Count by one. It stops forwarding data cells (only of this VC) when the Credit_Count reaches zero, and will be eligible to forward data cells again when receiving a new credit cell with a positive credit value (for this VC). When receiving a credit cell for a VC, the sender will immediately update its current Credit_Count for the VC using:

$$\text{Credit\_Count} = \text{Credit Value in the Newly Received Credit Cell} \quad (2)$$

Note that this is an absolute credit updating as defined in Section 3.

To prevent data overflow, the value of *N1* needs to be large enough so that the *N1* zone, as stated earlier, is able to catch all possible in-flight data cells. It is therefore necessary to know an upper bound on the number of these in-flight data cells, *F*. Note that the total amount of data in these data cells is no more than $R \cdot B_{link}$ [13]. This implies that $F \leq R \cdot B_{link} / Cell\_Size$. Therefore to prevent data overflow it is sufficient to choose *N1* to be[3]:

$$N1 = R \cdot B_{link} / Cell\_Size \quad (3)$$

To prevent data underflow, i.e., to ensure that the VC can deliver an average bandwidth of B_{VC} over time *R*, it suffices to choose *N3* to be:

$$N3 = R \cdot B_{VC} / Cell\_Size \quad (4)$$

By increasing the *N3* value, the VC can transport data cells at a proportionally higher bandwidth.

7. The *N123+* Scheme: Receiver-Enhanced, Credit-Based Flow Control Scheme

The "*N123+* Scheme" is the second credit-based scheme of this paper for implementing link-by-link flow control. By performing some additional work at the receiver, the *N123+* Scheme has the desirable property that the size of the VC buffer for a VC has an upper bound depending only on the targeted bandwidth of the VC (Equation (5)), not on the bandwidth of the underlying physical link (Equation (4)).

As depicted in Figure 7, the additional work at the receiver involves imposing a time gap of at least one round-trip link delay *R* between the sendings of any two consecutive credit cells for the same VC. A new credit cell of a VC can be sent only after *R* time has elapsed and at least *N2* data cells of the VC have departed since the previous cred-

[1] In all figures for describing the flow control schemes, the following notational conventions are assumed: time flows from top to bottom; the occupied area of a buffer is shaded; and dashed arrows refer to transmission of credit.

[2] As depicted in Figure 5, the *N1* zone is reserved to hold in-flight data cells. Thus it will not be used in calculating credit values in the *N123* and *N123+* Schemes.

[3] In general, it should be that $N1 = \lceil R \cdot B_{link} / Cell\_Size \rceil$, i.e., *N1* should be the ceiling integer on the quantity inside the bracket, or the smallest integer greater than or equal to the quantity. For presentation clarification, this paper assumes throughout that these qualities are integers so that the ceiling notation is not needed. It is straightforward to insert proper ceiling notions when this assumption is removed.

it cell for the same VC was sent. Refer to [13] for some implementation suggestion on ensuring this time gap.

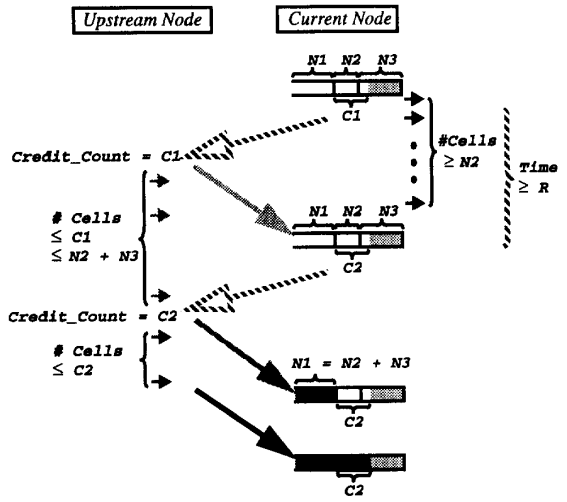

Figure 7: *N123+* Scheme and required *N1* value

By imposing a time gap of at least one round-trip link delay R between two consecutive credit cells, the *N123+* Scheme guarantees that any data cell arriving after the departure of the most recent credit cell must have reflected the credit value in this credit cell or in the previous credit cell, but in no earlier ones.

This implies that as depicted in Figure 7, those in-flight data cells which do not reflect the credit (value of *C2*) in the most recent credit cell must be no more than the credit (value of *C1*) in the previous credit cell. Since the credit value in any credit cell is not larger than $N2 + N3$, the number F of these in-flight data cells satisfy:

$$F \leq C1 \leq N2 + N3$$

where *N3* is given by Equation (4). This and Equation imply that to prevent data overflow it suffices to choose *N1* to be:

$$N1 = \min (N2 + N3,\ R \cdot B_{link} / Cell\_Size)$$

or

$$N1 = min\ (N2 + R \cdot B_{VC} / Cell\_Size,\ R \cdot B_{link} / Cell\_Size) \quad (5)$$

Since $N1 \leq N2 + N3$, the total buffer size, $N1 + N2 + N3$, for the VC is no more than $2 \cdot (N2 + N3)$, which is a quantity independent of the peak bandwidth of the physical link. When $N2 + N3$ is smaller than $R \cdot B_{link} / Cell\_Size$, the *N123+* scheme is more attractive than the *N123* scheme as far as minimizing the VC buffer is concerned.

8. The *N23* Scheme: Sender-Enhanced, Credit-Based Flow Control Scheme

The "*N23* Scheme" is the third credit-based scheme of this paper for implementing per VC link-by-link flow control. By performing some additional work at the sender i.e., Equation (7), the *N23* Scheme has a major advantage that the VC buffer for a VC does not need the *N1* zone.

Figure 8 depicts the basic algorithm for the *N23* Scheme, which is similar to that for the *N123* Scheme of Section 8. For completeness, a full description of the *N23* Scheme is nevertheless given below.

The current node is eligible to send a credit cell (to the upstream node) for a VC each time after it has forwarded *N2* data cells of the VC (to the downstream node) since the previous credit cell for the same VC was sent. The credit cell will contain a credit value (for the VC) equal to the number of unoccupied cell slots in the combined area consisting of the *N2* and *N3* zones.

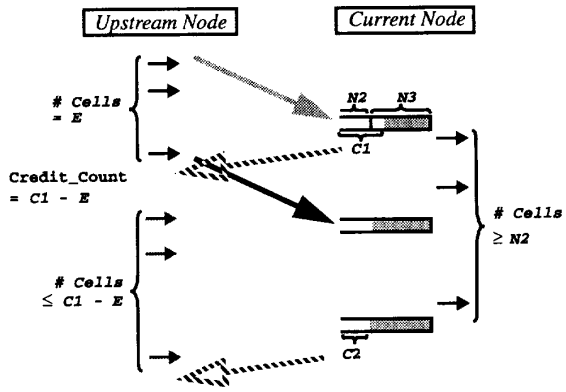

Figure 8: *N23* Scheme

Upon receiving a credit cell with credit value C for a VC, the upstream node is permitted to forward up to $C—E$ data cells of the VC before the next successfully transmitted credit cell for the VC is received, where E is defined by Equation (8). Specifically, the upstream node maintains a count, called Credit_Count, for the VC. Credit_Count could be set to be $N2 + N3$ initially. Each time the upstream node forwards a data cell of the VC (to the current node), it decrements the Credit_Count by one. It stops forwarding data cells (only of this VC) when the Credit_Count reaches zero, and will be eligible to forward data cells (of this VC) again when receiving a new credit cell (for this VC) resulting in a positive value for $C - E$.

More precisely, when receiving a credit cell for a VC, the sender will immediately update its current Credit_Count for the VC using:

$$\text{Credit\_Count} = \text{Credit Value in the Newly Received Credit Cell} - E \quad (6)$$

where

$$E = \#\text{ of data cells the sender has forwarded over the VC for the past time period of } R \quad (7)$$

Note that this updating is "absolute", as defined in the end of Section 3, as the new Credit_Count is computed independently of the old Credit_Count.

Like the N123 Scheme, it can be proven [13] that to prevent data and credit underflow so that a VC can sustain a targeted average bandwidth of B_{VC} over time R, that it suffices to choose $N3$ to be:

$$N3 = R \cdot B_{VC} / Cell\_Size \quad (8)$$

By increasing the $N3$ value, the VC can transport data cells at a proportionally higher bandwidth.

The *N23* Scheme can be viewed as an "ultimate credit scheme" in the sense that the sender is allowed to forward data cells against the current credit and sometimes also *against a credit yet to come*. When the credit arrives, the new credit value is immediately discounted by the amount the sender has already forwarded (i.e., the value of E) over the past round-trip link time R. Moreover, the *N23* Scheme is precise in the sense that when the current node issues a credit of value C for a VC, the node can expect to receive *exactly* C data cells of the VC, provided that the upstream node will have at least these many cells to forward.

8.1. Properties of the *N123* Scheme

P1 There is **no data overflow**, as long as corrupted credit cells can be detected by the 32-bit CRC in each credit cell [13].

P2 There is **no data underflow** and **no credit underflow** in sustaining a VC's targeted bandwidth as long as there are no corrupted credit cells. This means that when there are no hardware errors which corrupt credit cells, the VC never has to wait for data or credits due to the round-trip link delay associated with the flow control feedback loop. That is, the flow control mechanism itself will never prevent a VC from sustaining its targeted bandwidth.

P3 Corrupted credit cells, which are detected by the CRC and discarded, could cause some delay for the affected VC due to data or credit underflow, but no further harm. The delay is no more than, and can be much less than, the usual time-out period for recognizing errors plus the round-trip link delay required to recover from them. In fact, any possible effect of a corrupted credit cell will disappear immediately after the successful delivery of the next credit cell for the same VC. The receiver sends the next credit cell either automatically (i.e., after additional $N2$ cells have been forwarded, as described in Section 8) or in response to the sender's credit request (see the *sender-request-credit* transaction type in [13]). In this sense the flow control scheme is **robust** and **self-healing** (Section 8.8). Note that credit cells are "idempotent" with respect to the sender in that multiple receipts of credit cells, possibly including redundant ones, from the receiver will never cause harm.

P4 **Transmitting credit cells at any low bandwidth** (Section 8.7) is possible. By increasing the size of the VC buffer (i.e., the $N2$ value), the required bandwidth for transmitting credit cells decreases proportionally. This memory-bandwidth trade-off can be configured on a per VC basis.

P5 The absolute credit updating of Equation (7) is equivalent to the additive or increment credit updating (Section 8.2).

P6 **The size of the VC buffer for a flow-controlled VC has an upper bound depending only on the targeted bandwidth of the VC** (Equations and (5)), not on the bandwidth of the underlying physical link.

P7 **The average bandwidth achievable by a flow-controlled VC over time R is bounded above by $(N2 + N3) \cdot Cell\_Size / R$** (see [13]).

These properties and others are elaborated or proven in the next several subsections.

8.2. Some Mathematical Properties of the *N23* Scheme

Some interesting mathematical properties can be proven for the *N23* Scheme. Consider at the upstream node the arrivals of two consecutive, successfully transmitted credit values $C1$ and $C2$, for a VC, as depicted in Figure 9. Note that there could be any number of corrupted credit cells between $C1$ and $C2$ for the same VC, which the upstream node will discard. The figure depicts one such corrupted credit value, C'. A number of quantities need to be defined:.

- As E is defined in Equation (8), $E1$ or $E2$ is the number of data cells forwarded by the upstream node for the past time period of R immediately before the arrival of $C1$ or $C2$, respectively.
- If the set of data cells accounted for by $E1$ and that by $E2$ do not overlap (as in the case depicted by Figure 9), then x is defined to be the number of those data cells which are forwarded by the upstream node during the

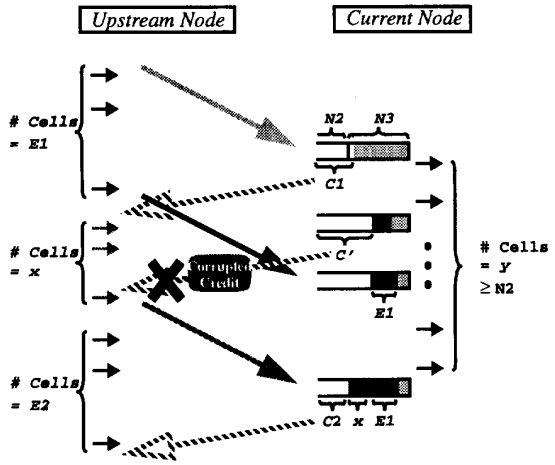

Figure 9: Definitions of various quantities for the *N23* Scheme, allowing the possibility of corrupted credit cells

period between the arrivals of *C1* and *C2*, and which are not accounted for by *E2*. On the other hand, if the set of data cells accounted for by *E1* overlap with that by *E2*, then x is the negative of the number of overlapping data cells.

- b (or the credit "balance" at the upstream node just before *C2* arrives) is the remaining Credit_Count when C2 arrives, that is to be overwritten by $C2 - E2$. The upstream node could have forwarded an additional b data cells even if *C2* had not arrived.
- y is the number of data cells the current node has forwarded to the downstream node during the period between the sendings of *C1* and *C2*.

An important property about the *N23* Scheme, Equation (10) below, can be derived by capturing what the method does at both the current and upstream nodes. At the current node when sending credit value *C2*, the following holds:

$$C1 = C2 + x + E1 - y$$

On the other hand, at the upstream node when receiving credit value *C2*, the following holds:

$$C1 = E1 + E2 + x + b$$

By eliminating *C1* from the above two equations, it follows that

$$C2 - E2 = b + y \tag{9}$$

Equation (10) says the following for each successfully transmitted credit cell for a VC. The discounted credit value $C - E$ (i.e., $C2 - E2$ in the equation) at the receiver is equal to the credit balance (i.e., b) for the VC plus the number (i.e., y) of data cells the sender has forwarded over the same VC during the period between the sendings of the previous successfully transmitted credit cell for the same VC and this current credit cell. The following conclusions hold:

- The new Credit_Count, $C - E$, computed by Equation (7), is always greater than or equal to *N2*, since $C - E = b + y$, $b \geq 0$, and $y \geq N2$.
- The new Credit_Count, $C - E$, computed by Equation (7), is exactly what would be computed by an additive or relative credit updating method [5, 6, 11]. The *N23* Scheme achieves this in an indirect and robust manner, *without* having to keep track of the quantity b at the sender and y at the receiver, and having to rely on successful transmission of each y value from the receiver to the sender.

8.3. No Data Overflow for the *N23* Scheme

The *N23* scheme has no data overflow for any values of *N2* and *N3*. The proof is based on the fact that

$$E = F \tag{10}$$

where, as defined in Section 4, F is the number of in-flight data cells which are not accounted for by the most recently received credit cell. Suppose that the credit value in this credit cell is equal to C. Then, as depicted in Figure 8, the VC buffer in the current node will have enough space to hold E in-flight data cells as well as C – E new data cells the sender will forward reflecting the C credit it will receive. Thus there is no data overflow.

8.4. No Data Underflow and Required *N3* Value for the *N23* Scheme

Figure 11 depicts that there is no data underflow for the *N23* Scheme when the value of *N3* is given by Equation (4).

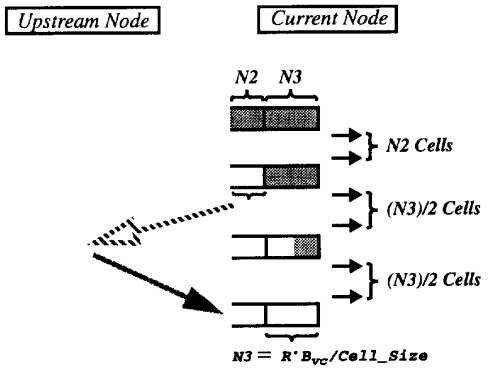

Figure 11: No data underflow for the *N23* Scheme and the required *N3* value

8.5. No Credit Underflow for the *N23* Scheme

Figure 12 shows that there is no credit underflow for the *N23* Scheme, assuming that the VC operates at the constant rate of B_{VC} and *N3* is given by Equation (4). Note that at the beginning of each repetitive period, the upstream node will start with a new Credit_Count equal to *N2*, which is the just received credit value *N2* + *N3* less *N3*, the number of data cells forwarded in the past time period of *R*. Since both the upstream and current nodes are assumed to operate at the same rate B_{VC}, at the end of the repetitive period when the upstream node has forwarded *N2* data cells and has depleted its Credit_Count, the next credit cell with credit value equal to *N2* + *N3* will arrive. Thus the upstream node will never have to stop forwarding data cells because its Credit_Count has been depleted.

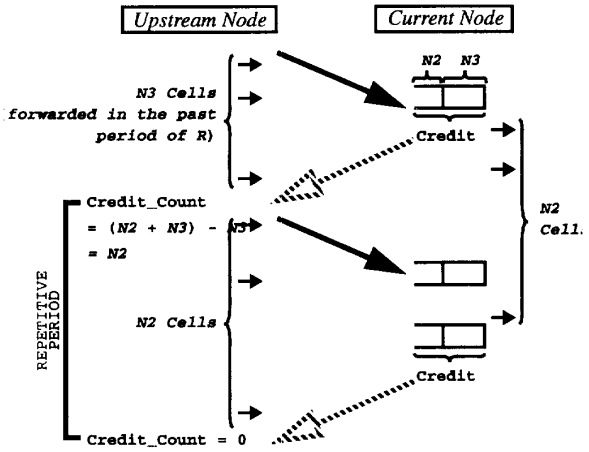

Figure 12: No credit underflow for the *N23* Scheme

8.6. Bounding the Bandwidth Realizable by a VC for the *N23* Scheme

The *N23* Scheme guarantees that over *any* time interval of length *R* the upstream node forwards no more than *N2* + *N3* data cells for the VC and thus:

$$B_{VC} \leq (N2 + N3) \cdot Cell\_Size / R \qquad (11)$$

To prove this, assume without loss of generality that such an interval, called the *R*-Interval, ends after the arrival of the *i*-th credit cell and before or on the arrival of the (*i*+1)-th credit cell for some *i*, as depicted in Figure 13. Since the *R*-interval has length no more than *R*, it must be entirely contained within the concatenation of the following two subintervals: (a) time period of length *R* preceding the arrival of the *i*-th credit cell, and (b) time period between the arrival times of the *i*-th and (*i*+1)-th credit cells. Note that the upstream node forwards *E* data cells during subinterval (a) by the definition of *E*, and no more than *N2* + *N3* – *E* data cells during subinterval (b) by Equation (7). Therefore the upstream node forwards a total of no more than *N2* + *N3* data cells during the two subintervals and thus during the *R*-interval.

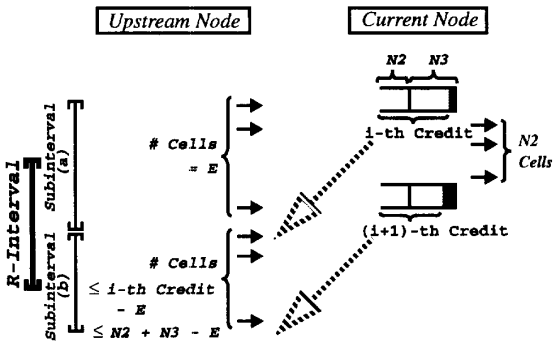

Figure 13: An upper bound, (*N2* + *N3*), on the maximum number of data cells that the upstream node can forward over any time interval of length *R* in the *N23* Scheme

8.7. Bounding the Bandwidth for Transmitting Credit Cells and Required *N2* Value

For a given value of *N2*, the current node will send a credit cell for the VC to the upstream node each time after having forwarded *N2* data cells of the VC to the downstream node. Thus over this VC, the link between the current and upstream node will transport credit cells no more than once every *N2* data cells. By using *N2* value of at least *X* for all VCs, the overhead of transmitting credit cells can be limited to an arbitrary fraction (1/*X*) of the link bandwidth. However, the larger the value of *N2* is, the larger the required memory in the *N2* zone is. The selection of the *N2* value is a design or engineering choice. *N2* can be chosen to be about 10 so that credit cells consume no more that about 10% of total network bandwidth.

8.8. Robustness of the *N23* Scheme

Using a strong error check such as a 32-bit CRC (see a proposed credit cell format including CRC-32 in [13]), the probability of undetected incorrect credit cells can be kept at an acceptably low level. A corrupted credit cell detected by the CRC at the sender will be discarded and the arrival of the next successfully transmitted credit cell for the same VC will recover from the error automatically.

After the sender detects and discards a corrupted credit cell, let *A* be the number of future credit cells that will ar-

rive anyhow before the next successfully transmitted credit cell is received. Then $A = \lfloor (B + G) / N2 \rfloor$ where B is the number of data cells of the VC in the receiver at the sending time of the credit cell (which becomes corrupted), and G is the number of additional data cells of the VC which may still arrive reflecting the last successfully delivered credit cell. Note that $B + G$ can be as large as the maximum number of cells the VC buffer can hold.

Thus after the sender discards a corrupted credit cell, there could be A additional future credit cells for the same VC forthcoming, any of which if successfully transmitted, will remove any effect caused by the corrupted cell. The value of A can be increased to improve this automatic protection against corrupted credit cells, by decreasing $N2$ (at the expense of increased bandwidth overhead for credit cells transmission) or increasing the VC buffer size, or both. Note that networks with large propagation delays have larger VC buffers (because of a large value for $N3$) and therefore the credit-based flow control schemes using absolute updating will be more fault-tolerant than networks with small propagation delays.

However, after the sender detects a corrupted credit cell, sometimes another credit cell will not immediately follow after the corrupted credit cell, because $B + G < N2$. In other words, the corrupted credit cell is the last credit cell generated by a burst of data. In this case, after the sender has waited for credit cells for the VC longer than some time-out period, it can request the receiver to send credit value for a VC (using, for example, *Sender-Request-Credit* in [13]). The receiver can also send redundant credit cells for a VC at any frequency to increase the protection against error. The absolute nature of the credit count updating by Equation (7) is such that as discussed in Section 3, the effect of lost credit cells can be automatically recovered by any future successfully transmitted credit cell for the same VC. The only impact of a lost credit will be a potential, temporary delay in forwarding more traffic over the VC.

8.9. Orthogonal Relation to Switching and Scheduling

The credit-based per VC flow control mechanisms discussed in this paper are orthogonal to issues related to switching and scheduling functions associated with a switching node. For a flow-controlled VC link, one side of the link does not need to know whether the other side is a switch or not. The flow control itself has no concerns on implementation matters at either side of the link related to how scheduling and/or switching of data cells of various VCs are performed.

That is, flow control functions prevent data overflow and underflow, whereas switching and scheduling functions are responsible for implementing various services, such as guaranteed bandwidth and latency for certain VCs, on top of the flow control mechanism.. As discussed in [13], underlying flow control schemes can facilitate efficient scheduling, namely, a scheduler need only concern itself with VCs with data and credit. For example, sophisticated scheduling algorithms such as various fair queueing methods [7, 16] can be implemented on top of flow control, completely unburdened by buffer management issues

9. Summary of Three Credit-Based Flow Control Schemes

Figure 14 summarizes the three credit-based flow control schemes described above and their VC buffer sizes. Below are some comments on $N1$, $N2$ and $N3$ values:

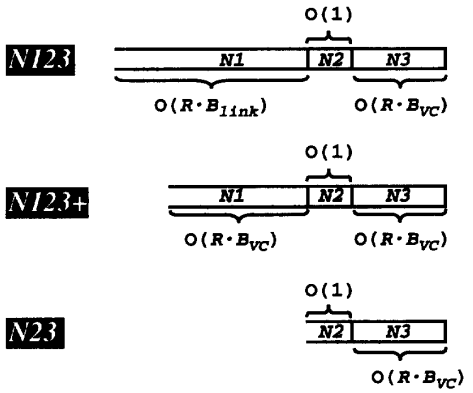

Figure 14: Three credit-based flow control schemes of this paper, and their VC buffer sizes

- Figure 14 depicts the case that the $N1$ zone of the $N123+$ Scheme is smaller than that of the $N123$ Scheme. This holds when $B_{VC} < B_{link}$ and R is sufficiently large.
- For all the three methods, $N2$ is a constant, independent of R, B_{VC} or B_{link}. For example, $N2$ can be chosen to be 10 for all VCs so that transmission of credit cells will consume no more than 10% of the network bandwidth. As noted in Section 5, different $N2$ values can also be used for different VCs.
- For a VC with a targeted bandwidth of B_{VC}, $N3$ is chosen according to Equation (4), i.e.,

  $N3 = R \cdot B_{VC} / Cell\_Size$

  When the value of $N3$ is so chosen, the VC will have sufficient buffer space to sustain bandwidth as high as B_{VC}, as shown above.

10. VC Buffer Sizes for the *N23* Scheme

Figure 15 is a summary of the required per VC buffer sizes for the *N23* Scheme at each node, for various link lengths and targeted VC bandwidths B_{VC}. The calculation assumes that *N2* = 10 for all VCs, and the propagation delay per km is 5 microseconds. In addition, it assumes that the time for handling data cells and processing credit cells at the two endpoints consumes additional 5 microseconds, as in the BNR/Harvard ATM switch.

	VC Bandwdith (B_{VC})		
	10 Mbps	**100 Mbps**	**622 Mbps**
Link Length			
1 km	10+1 cells 1 KBytes	10+4 cells 1 KBytes	10+23 cells 2 KBytes
10 km	10+3 cells 1 KBytes	10+25 cells 2 KBytes	10+155 cells 9 KBytes
100 km	10+24 cells 2 KBytes	10+238 cells 14 KBytes	10+1475 cells 79 KBytes
1000 km	10+235 cells 13 KBytes	10+2360 cells 126 KBytes	10+14679 cells 779 KBytes

Figure 15: Per VC buffer size in #cells (i.e., *N2* + *N3*), and #KBytes at each node for *N23* Scheme

11. Full Link Utilization: "Overbooking Inequality"

One can see from Figure 15 that local area networks can support a large number of VCs with large B_{VC}, using only a moderate amount of memory. Each of these VCs may assume a high bandwidth at various times whenever network load permits. For example, on a 1 km link, one megabyte of memory can maintain 1,000 VCs, each of which can operate at a speed as high as 100 Mbps, i.e., at B_{VC} = 100 Mbps.

These VCs obviously cannot all operate at their peak bandwidth simultaneously, over the same physical link of bandwidth B_{link}, equal to say, hundreds of megabits per second. That is, the "overbooking" inequality holds:

$$\sum B_{VC} > B_{link} \qquad (12)$$

where the summation is over all the VCs on the link. In fact, the idea of LLFC is to make the left-hand side much larger than the right-hand side, in order to maximize link utilization. The credit-based flow control schemes of this paper will flow control these VCs dynamically so that they can slow down when the link is congested. However, as soon as the link congestion situation lightens, each of these VCs can immediately operate at speeds as high as possible, up to its peak bandwidth B_{VC}, to make the maximum-possible use of the available bandwidth.

In some sense, the fundamental reason to use link-by-link flow control or fast feedback is to allow the "overbooking inequality" (13), in order to let VCs peak at high speeds to make efficient use of link bandwdith slacks as soon as they become available. This is in sharp contrast with contract-based, rate-control approaches through which the VC admission process will disallow such inequality.

Figure 16 shows an example of this overbooking. *VC1* has high priority and alternates between 10% and 66% of the link bandwidth; it might carry video. *VC2* is greedy and low priority; it might be a file transfer. The two VCs share a link that has a round-trip time of 155 cell times. Each VC has an *N3* large enough to send at the full link rate; by Equation (2) this is 155. Of course, they cannot both send at that rate, and whenever there is competition, *VC1* will win. Ideally, *VC2* would vary the rate at which it sent so as to use all bandwidth not used by *VC1*, keeping the link fully utilized.

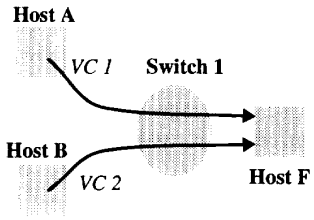

Figure 16: Full Link Utilization

The results of a simulation of this scenario in [14] are shown in Figure 17. The throughputs of the two VCs always sum to 100%, so *VC2* is in fact filling the gaps in *VC1*'s traffic. Furthermore, *VC2* is doing this without buffering large numbers of cells in the switch: *VC2* never uses more than *N2* + *N3* = 10 + 155 = 165 cell buffers. Notice that the graph of *VC2* lags slightly behind *VC1*, as is expected. Both of these result from the flow control mechanism's ability to quickly increase or decrease a VC's throughput as conditions change in the network. See [14] for the detailed behavior of this mechanism.

The simulator that produced the results of Figure 16 and others in [14] was designed primarily to verify the architecture of the BNR/Harvard switch. It models much of the switch down to the level of registers and clock cycles, and thus provides very accurate timings at the level of individual ATM cells.

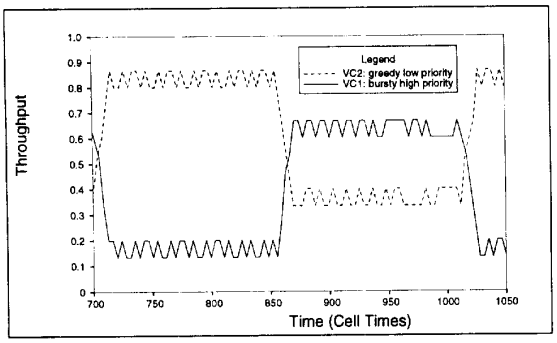

Figure 17: The greedy VC (VC2) takes up the left over bandwidth. The link is fully utilized.

12. Concluding Remarks

The credit-based flow control schemes proposed in this paper can support "best-effort" traffic well, which unlike guaranteed traffic, is not amendable to control without feedback. For a network where both best-effort and guaranteed traffic are present, shaping at network input together with scheduling without feedback at each node, can be used to control the component of admitted guaranteed traffic [16]. In the meantime per VC link-by-link flow control can be used to control other traffic. As described in Section 11, low-priority best-effort traffic can dynamically fill the bandwidth gap possibly left by the guaranteed traffic, thus achieving full or high link utilization. "Greedy" services, as defined in Section 2, can therefore be implemented.

The proposed flow control schemes are efficient and robust. Simulation results [14] have confirmed this. When network congestion occurs, the size of the VC buffer of a flow-controlled VC need never grow beyond a predetermined limit related to the VC's desired bandwidth and the round-trip link delay. In general flow-controlled VCs use much smaller peak buffer sizes than non flow-controlled ones while delivering the same throughput. The flow control mechanism itself does not create extra delays under uncongested situations. If congestion does occur because the size of the buffer of a flow-controlled VC is limited, the delay will be bounded when the congestion clears up.

References

[1] ATM Forum, "ATM User-Network Interface Specification," Version 2.1, May 1993.

[2] ANSI T1S1.5 "AAL5 - A New High Speed Data Transfer AAL," Stds. Proj. T1S1.5 AAL-ATM, Nov. 4-8, 1991.

[3] "Preliminary Report on Broadband ISDN Transfer Protocols," Bellcore Special Report, SR-NWT-001763, Issue 1, Dec. 1990.

[4] "High-Performance Parallel Interface - Mechanical, Electrical and Signalling Protocol Specification (HIPPI-PH)" , ANSI X3.183-1991.

[5] S. Borkar, R. Cohn, G. Cox, S. Gleason, T. Gross, H. T. Kung, M. Lam, B. Moore, C. Peterson, J. Pieper, L. Rankin, P. S. Tseng, J. Sutton, J. Urbanski, and J. A. Webb, "iWarp: An Integrated Solution to High-Speed Parallel Computing," *Proceedings of Supercomputing '88 Conference*, Orlando, Florida, November 1988, pp. 330-339.

[6] 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.

[7] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Proc. SIGCOMM '89 Symposium on Communications Architectures and Protocols*, pp.1-12.

[8] H. J. Fowler and W. E. Leland, "Local Area Network Traffic Characteristics, with Implications for Broadband Network Congestion Management," *IEEE J. on Selected Areas in Commun.*, vol. 9, no. 7, pp. 1139-1149, Sep. 1991.

[9] M. W. Garrett, "Statistical Analysis of a Long Trace of Variable Bit Rate Video Traffic," Chapter IV of Ph.D. Thesis, Columbia University, 1993.

[10] D. Grunwald, "A Users Guide to Awesime: An Object Oriented Parallel Programming and Simulation System," University of Colorado at Boulder Technical Report CU-CS-552-91, 1991.

[11] 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.

[12] H. T. Kung, "Gigabit Local Area Networks: A Systems Perspective," IEEE Communications Magazine, 30 (1992), pp. 79-89.

[13] H. T. Kung and A. Chapman, "The FCVC (Flow-Controlled Virtual Channels) Proposal for ATM Networks," Version 1.1, 1993.

[14] H.T. Kung, R. Morris, T. Charuhas, and D. Lin, "Use of Link-by-Link Flow Control in Maximizing ATM Networks Performance: Simulation Results," *Proc. IEEE Hot Interconnects Symposium, '93* Palo Alto, California, Aug. 1993.

[15] W. E. Leland, M. S. Taqqu, W. Wilinger and D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic," *Proc. SIGCOMM '93 Symposium on Communications Architectures and Protocols*, 1993.

[16] A. Parekh and R. G. Gallager, "Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks - The Multiple Node Case," *IEEE INFOCOM '93*, San Francisco, Mar. 1993.