A High Speed Implementation of Adaptive Shaping for Dynamic Bandwidth Allocation

Cameron Braun, Vinai Sirkay, Hugo Uriona, Srini Seetharam, Esmaell Yousefi
David Petr, Douglas Niehaus, Victor Frost, Joseph Evans, Gary Minden

Telecommunications & Information Sciences Laboratory
Department of Electrical Engineering & Computer Science
Lawrence KS 66045-2228

Abstract

Most algorithms proposed for controlling traffic prior to entering ATM networks are based on static mechanisms. Such static control mechanisms do not account for the dynamics of the user traffic or the network state. Some dynamic control algorithms have been proposed, but most of these algorithms are extremely complex and may make it difficult to provide real time control. In this paper, we present an adaptive rate control algorithm that has been implemented in hardware. The algorithm controls the traffic submitted by a source based on the indirectly observed average rate and burst size for the source. The algorithm is highly efficient and thereby provides real time control at high speed. Our implementation, in concert with flow control in the local area network, provides the basis for ATM-based high performance distributed systems.

1: Introduction

The future of telecommunication networks is one of seemingly limitless resources and countless applications and users. The reality in Asynchronous Transfer Mode (ATM) networks, however, is that customers can consume network resources thereby congesting networks if no constraints are in place. One such resource that needs to be managed is the bandwidth for a particular customer. Bandwidth management can be divided into three categories: bandwidth reservation, bandwidth limitation and bandwidth allocation. While these three terms are often thought to be the same, there are important differences.

Bandwidth *reservation* dedicates bandwidth to a user such that even if only a fraction of the reserved bandwidth is utilized, the remaining portion of the reserved bandwidth is not available to any other users in the network. A bandwidth *limitation* constrains the maximum amount of traffic that can be sent by a single user. If the user attempts to send more traffic than the upper limit allows, the traffic could be discarded, buffered, or otherwise penalized because it does not conform to the specified limits. The third area of bandwidth management, *allocation* (as we use the term), is similar to bandwidth reservation in that the bandwidth is "guaranteed" to be available to the customer; it differs in that if the customer does not use all of the allocated bandwidth, the unused portion is made available to others.

Bandwidth management may be applied on the customer side or the network side of the ATM User-Network Interface (UNI) [6]. For example, customers are encouraged to produce traffic that does not violate the traffic contract with the network provider. The network side may *police* the customer traffic and penalize (discard and/or delay) nonconforming traffic. To produce conforming traffic the customer may *shape* the traffic between the source and the network. Shaping or smoothing refers to queueing ATM cells and then releasing those cells so that the burstiness of the source is controlled.

While there are many different methods of bandwidth management they all fall into one of two categories: static or dynamic. Static resource management approaches as suggested in [1, 4, 8] do not reflect the inherent dynamic nature of user requirements and the network state, which may change during the lifetime of the connection. Specifically consider the case where n virtual circuits (VC's) share a transmission facility, either a physical or a virtual path (VP). The QoS for the case with the dynamic adjustment of the resources among the n VC's is superior to that obtained by static assignment schemes.

Dynamic bandwidth allocation changes the bandwidth allocated to a particular customer over time. This is done by determining the behavior of customer traffic and allocating available network resources to accommodate the traffic. Customer bandwidth usage characteristics can be determined by monitoring the submitted traffic. An important feature of dynamic bandwidth allocation is the ability to make bandwidth changes based on continuous monitoring of customer traffic. If the customer increases the amount of traffic being sent, the algorithm should allocate more network resources, and if the customer reduces the traffic submitted to the network, the algorithm should reduce the amount of network resources that had been allocated to the particular customer [2]. Since the customer is typically unable to accurately characterize the traffic, we propose a scheme that can dynamically change its parameters to follow the trends of the customer's traffic and allocate bandwidth according to its usage. This paper specifically describes an adaptive I-out-of-M shaping algorithm that would be part of this overall dynamic bandwidth allocation scheme.

The I-out-of-M sliding-window algorithm provides a mechanism to control the average throughput and burst characteristics of a virtual circuit (VC)[6]. The control for this algorithm is based on the usage and return of credits which are required for a VC to be able to transmit. If a

The authors can be contacted via e-mail at evans@tisl.ukans.edu.
This research is partially supported by ARPA under contract F19628-92-C-0080, Digital Equipment Corporation, the Kansas Technology Enterprise Corporation, and Sprint.

VC runs out of credits, its traffic is queued and it will not be allowed to transmit until a credit is returned. This algorithm can transmit a maximum of *I* ATM cells in any *M* cell slot interval, yielding a maximum sustained throughput of *i/m* normalized to the line rate. The value of *I* also indicates the maximum number of consecutive cells that can be transmitted. As a VC transmits a cell, the credit bank for that VC is decremented by one and that credit is returned to the VC after *M* cell slots. This algorithm is suitable for traffic that is transported in a slotted ATM over SONET environment. Different VCs may have different values of *I* and *M* to provide more flexibility in control. The adaptive I-out-of-M shaper will change the *I* and *M* values according to the measurements obtained in order to match the rate (*I*/*M*) and burst (*I*) characteristics of the source [2]. We will show that the proposed adaptive shaping algorithm is superior to static shaping and that it is simple enough to be implemented at OC-12c rates. This is in contrast with previous algorithms for dynamic traffic control such as [7, 5, 10, 3, 16, 9] that are extremely complex to implement in hardware, thereby making real time control more difficult.

The I-out-of-M algorithm is being developed as a part of the Multidimensional Applications and Gigabit Internetwork Consortium project (MAGIC) [12]. The DEC AN2 ATM switch being used for this work has two different types of line cards. The first type is a host line card that has four full duplex links operating at 155 Mb/s, and the other type of line card is called the AN2/SONET Gateway which operates at 622 Mb/s using the SONET transmission protocol [11]. The AN2/SONET Gateway is the line card that implements the proposed algorithm and supports operation at the SONET STS-12/STS-12c (622.08 Mb/s) rate.

The paper is organized as follows: in Section 2:, a detailed discussion of the proposed control algorithm is presented; in Section 3: we discuss the simulations performed and the results obtained. In Section 4: a hardware/software implementation of the system is provided; finally we conclude by summarizing the paper in Section 5:.

2: The Adaptive Shaping Algorithm

2.1: Measurement Parameters

An obvious approach to adaptive traffic control would be to monitor the source output and then adapt the control algorithm according to the measurements of the source. However, it is not possible to directly monitor the source output on the AN2/SONET Gateway; only the output of the round robin queue on the gateway card, after the I-out-of-M shaper, can be observed (see Figure 1). The traffic characteristics of the output of the round robin queue differ from the source due to queuing effects, the contention arbitration performed by the DEC AN2, and the I-out-of-M shaping. These changes to the source traffic stream make it difficult to develop measurements that would give an accurate estimate of the source traffic. It was found from experimentation that the average rate and average burst parameters best capture the actual source outputs characteristics and are suitable for I-out-of-M adaptation [2].

2.1.1: Average Rate Parameter

A running count of the number of cells transmitted on a particular VC in a known measurement interval can be used to compute the average rate for the VC. The average rate

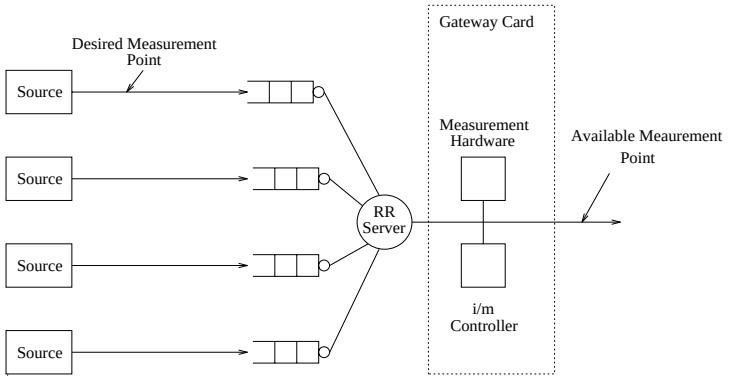

Figure 1. Customer Premise Equipment showing measurement points

is computed by dividing the number of cells transmitted by the time interval during which the measurements where gathered, that is,

$$\lambda = \frac{\text{total number of cells in a measurement period}}{\text{measurement interval}}.$$

The hardware implementation is described in Section 4:.

2.1.2: Average Burst Size Parameter

The shaping measurement hardware also provides the number of bursts that have occurred on a given VC in the measurement interval. This can be used in conjunction with the cell count described above to yield the average number of cells in each burst:

$$\beta = \frac{\text{total number of cells in a measurement period}}{\text{number of bursts in a measurement interval}}.$$

For a burst to occur the line must initially be idle (no cells from any VC are being transmitted) then become active (at least one cell transmitted on any VC), and again go idle. If a VC transmits at least 1 cell during a burst, then the burst count for that VC is incremented.

2.2: Control Mechanism

Now that the measurement variables have been described, the actual algorithm can be presented (see Figure 2). The values for *I*, *M*, and the measurement interval must be initially set by the customer. Once a measurement interval is over, the dynamic control process receives the raw measurements that are taken by the hardware and calculates the rate and average burst size as described above. Rate utilization is calculated as:

$$r = \frac{\lambda}{I/M}.$$

This parameter measures how closely the shaper is estimating the actual rate of the source. At this point, the algorithm begins the adaptation on the burst parameter by comparing the value of *I* to the average burst size. If *I* is less than $1.1 * \beta$ then *I* must be increased (the number of credits is too small). Similarly, if *I* is greater than $1.3 * \beta$ then *I* must decrease (the number of credits is too large). If *I* is between 1.1 and 1.3 times the value of β no adaptation is performed on *I*. Once the direction for the adaptation has been determined, the *I* adaptation is:

$$I = I + (I * \mu +)$$

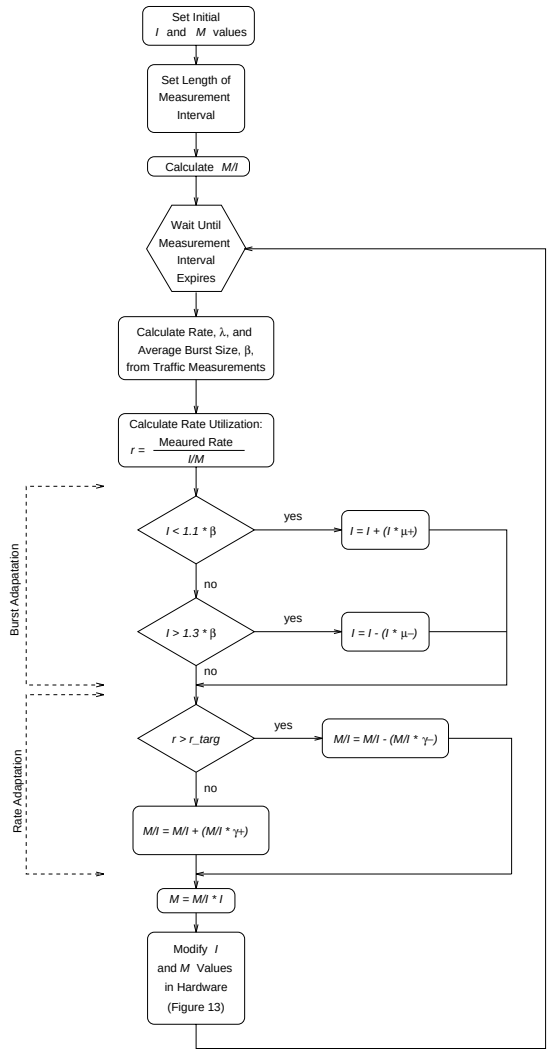

Figure 2. Adaptive I-out-of-M shaping algorithm

for an increase in the number of credits, or

$$I = I - (I * \mu-)$$

for a decrease in the number of credits. The parameters μ+ and μ- are adaptation factors. The values of 1.1 and 1.3 for the hysteresis in the *I* comparison were chosen after performing test simulations.

The decision for the direction of the rate adaptation is simpler because there is no hysteresis. If the rate utilization, *r*, is greater than a threshold, *r_targ*, the rate (*I/M*) is increased and otherwise the rate is decreased. For the simulations presented in section 3, *r_targ* = 0.85, meaning that the target shaper rate will be 15% greater than the computed average rate, λ. The parameters γ+ and γ- are the adaptive coefficients for *M/I*. It should be pointed out that the calculation of *M* divided by *I* leads to the inverse of the rate (slots/cell). The reason for this calculation is that the dynamic control process can perform a simple multiplication to arrive at *M*

$$M = M/I * I.$$

Hence, the rate adaptation is performed according to the equations

$$M/I = M/I - (M/I * \gamma-)$$

for an increase in the shaper rate, or

$$M/I = M/I + (M/I * \gamma+)$$

for a decrease in the shaper rate.

3: Simulation Results

The most obvious method to test an adaptive algorithm of any kind is to use a step function or a pulse as the input to the system. In the case of the dynamic i/m algorithm there are two adaptive parameters that must be tested, *I* and *M*. Since it would be most informative to test the two adaptations separately, the traffic source parameters were adjusted so that only one of the parameters was tested during a single simulation. Figure 3 shows the tracking ability of the algorithm for a pulse in the source rate, with γ+ = γ- = 0.2. It is obvious that the algorithm reacts to the source rate pulse, but it is also evident that the reaction time is slow resulting in queue build up and hence delay and possible cell loss. This delay problem can be addressed by selecting adaptation constants, γ+ and γ-, for the rate that will result in a faster attack. However, due to the abrupt changes in the source there will still be a delay spike because the algorithm cannot adapt to the changes instantly. Again only one of the parameters was tested at a time and Figure 4 shows that the source burst size was kept relatively constant during the rate increase. There is some noise in the response for the burst parameter, *I*, for the first 3000 slots, during which time the algorithm is attempting to adapt both the rate and burst parameters. As the input stimulus (the rate bump) persists, the algorithm eventually adapts only the rate parameter.

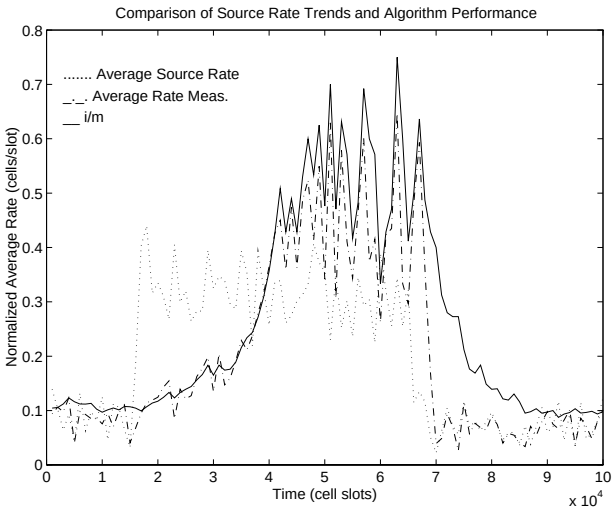

Figure 3. Adaptive algorithm tracking a change in the source rate

Figure 5 shows the ability of the algorithm to track a change in the burst size of the source, with the source rate held constant. The source burst size is increased from an average of 10 cells to 50 cells for 50,000 slot times (15,000

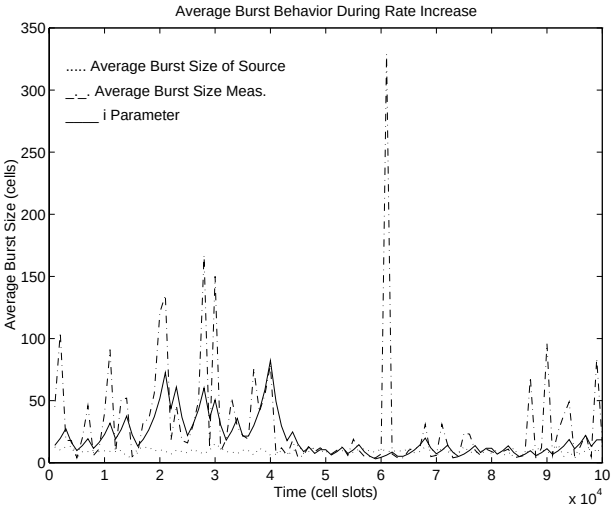

Figure 4. Adaptive algorithm smoothing noise in the source burst size

	Time in Cell Slots		
	0 - 15000	15000 - 65000	65000 - 100000
Average Source Burst Size	9.13	47.88	11.22
Average I	18.29	24.89	19.27

Table 1. Comparison of Average Source Burst Size and Average Burst Parameter

- 65,000). While the increase in the source rate is clear from Figure 5, the average burst measurement, λ, and the average burst parameter, I are not that clear from the figure. Table 1 shows the average source rate and the average I for the periods before, during, and after the burst bump. These numbers show that the source rate increases by a factor of five and that the average burst parameter, I does follow the source burst size, but not as close as the rate adaptation shown in Figure 3. This behavior for I is not unexpected since rate is simply the number of cells seen in a time interval, but it is harder to develop an accurate measure for the average burst size when observing only the shaped traffic stream. The rate of attack for I can be changed by changing the adaptation factors, $\mu+$ or $\mu-$. The adaptation shown here is for $\mu+ = \mu- = 0.4$ so a value of 0.5 or 0.6 would shorten the time in which I reacts. It is also possible to set the adaptation factors so that I rises faster than it falls, i.e. $\mu+ > \mu-$. Since there was no change in the source rate, there should be no change in the shaper rate, I/M, (see Figure 6). There is noise in the estimated source rate, but no appreciable change, and both the shaper rate and the measured rate track the source rate.

The previous results have shown that the algorithm can adapt to changes in the source traffic characteristics, but there needs to be a benefit for the increase in hardware. Table 2 shows why the algorithm is beneficial and why it makes sense to increase the size and complexity of the transmitting system. The graph compares the long-term average cell delay for the adaptive I-out-of-M algorithm

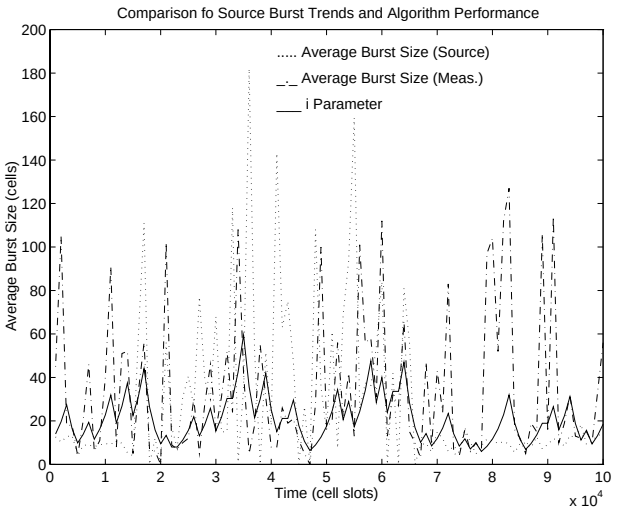

Figure 5. Adaptive algorithm tracking a change in the source burst size

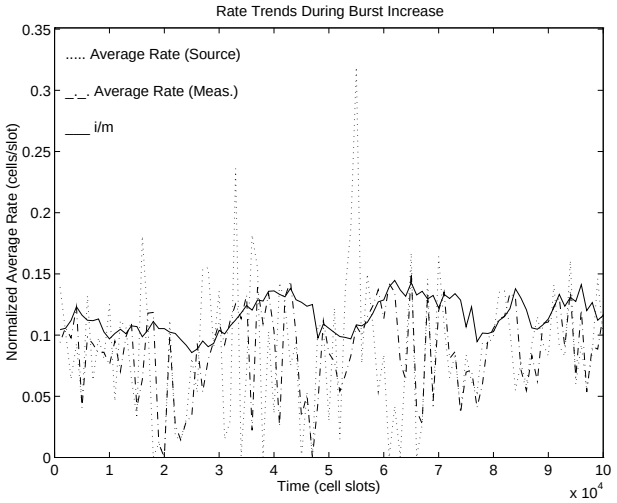

Figure 6. Adaptive algorithm smoothing noise in source rate

	Fixed Traffic (Average Burst = 30, ρ = 0.1)				
	Dynamic I-out-of-M	Static I-out-of-M			
	Average I = 20.85	10	20	30	40
Delay (slots)	661	904	874	844	816

Table 2. Delay Comparison for Adaptive and Non-Adaptive I-out-of-M Shapers

with static I-out-of-M shapers that have various values of *I*, but the same *I/M*. The source behavior for all these simulations is the same as for the burst bump described above. The adaptive I-out-of-M shaper can achieve lower delays than the static I-out-of-M shaper even for static *I* values that are larger than the average *I* for the adaptive case. By increasing the value of *I*, in the static case, it would be possible to achieve delays that are lower than those for the adaptive shaper (these larger static *I* results are not shown). However, there is an assumed cost for the burst size and in the case of the "burst bump" (changing source burst size while keeping the rate constant) presented here, the larger burst size will not be fully utilized when the source burst size is smaller than the static burst size value. The user would be paying for the burst size and not using it fully, whereas the adaptive algorithm would use a larger burst size only where the customer needs it and therefore would save the customer money and would save resources for the network provider.

4: Implementation

This section discusses the hardware - software implementation of the adaptive traffic control algorithm described in Section 2:. The algorithm uses two Field Programmable Gate Array (FPGA) components, the I-out-of-M controller and the shaping measurement hardware, and a general purpose processor [14].

The general purpose processor is called the Line Card Processor (LCP) and is a MIPS R3000 RISC processor with an embedded preemptive priority multitasking real-time operating system called DECelx. One task performed by the LCP is the periodic calculations that are involved in computing the new *I* and *M* values as described in Section 2.2:.

Figure 7 provides a view of all the components of the system that are involved in the implementation of the I-out-of-M control algorithm. The functionality and design of the I-out-of-M and shaping measurements hardware units are disussed in sections 4.1: and 4.2: respectively. The hardware can adaptively control a maximum of 127 VC's [14].

4.1: I-out-of-M Controller

The I-out-of-M controller can be divided into two functional blocks, the flow control hardware mechanism and the credit return mechanism. The flow control hardware unit is responsible for rate controlling the VC's based on credits used by the VC. The credit return mechanism is the open loop credit control mechanism for the flow control hardware unit.

The I-out-of-M controller manages several memory structures which contain information on current bandwidth

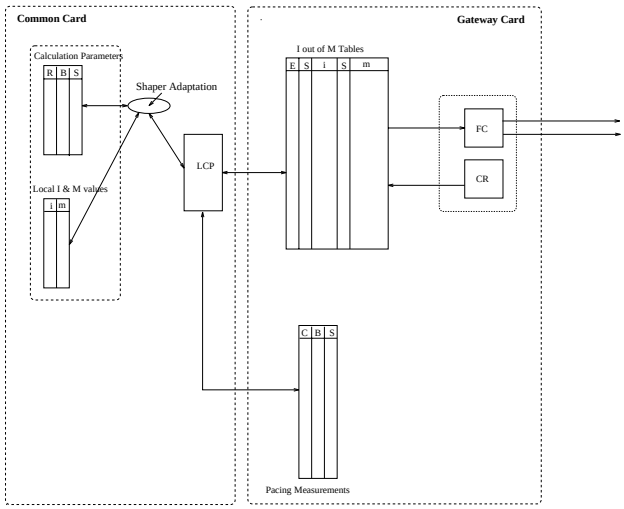

Figure 7. System Block Diagram

usage relative to allocated bandwidth for each VC. Figure 8 shows the I-out-of-M controller and its interaction with the memory structures.

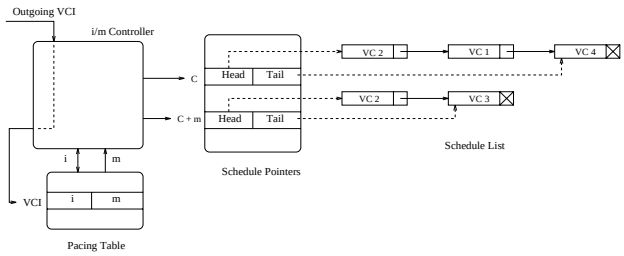

Figure 8. I-out-of-M Block Diagram

4.1.1: Shaping Table

The Shaping Table contains the information needed to control the bandwidth of the VC. This table is indexed by Virtual Circuit Identifier (VCI). For each VCI the table contains a 10 bit signed *i* value, which is the number of credits in the VC account, an 11 bit unsigned *M* value, which is the window size in cell slots, a flag to indicate if the VC is bandwidth controlled or not, and a semaphore bit that indicates to the credit return hardware the modification of a VC's *I* and *M* values by the LCP (see Section 4.3:). The current size of the Shaping Table is 4096 entries [13, 14].

4.1.2: Schedule Pointers Table

The Schedule Pointers Table keeps track of the pointers for the queues of VCs that are scheduled for credit update at different cell slots. Each entry in this table consists of a 16 bit Head pointer and a 16-bit Tail pointer, which are used as indices to the Schedule List. The most significant bit of each pointer is a nil indicator. The current implementation supports 2048 entries which is the maximum allowed window size (in cell slots) [13, 14]. Figure 9 shows the structure of the schedule pointer table.

4.1.3: Schedule List

The VC queues are organized in the Schedule List as linked lists. Each element in the list contains the following:

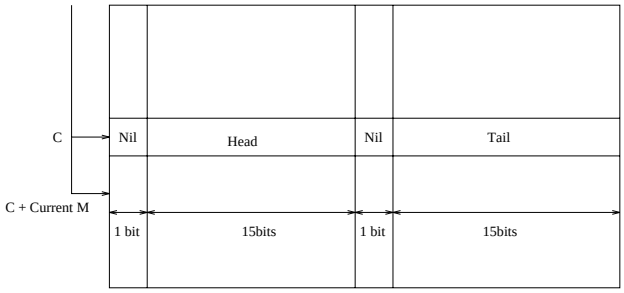

Figure 9. Schedule Pointers Table

a 12 bit VCI value, whose credit needs to be updated; a 16 bit pointer to the next element in the queue; and a nil flag to indicate the end of the list. The current size of the Schedule List is 16384 entries [14]. Figure 10 shows the schedule list memory structure.

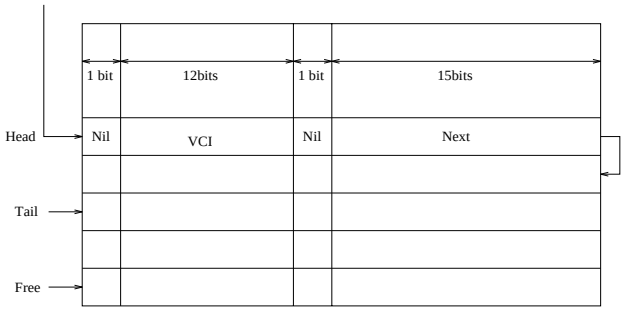

Figure 10. Schedule List

4.1.4: *Flow Control Hardware*

The flow control hardware is a FPGA that decrements the value of *i* in the shaping table of a VC after a cell is transmitted on that VC and schedules a return of this credit *M* slots later. When a cell is transmitted on a particular VC, the flow control hardware uses the VCI to index the Shaping Table and read the *i* and *M* values. The *i* value is then decremented by one [14].

If the new *i* value is less than a threshold value, meaning that the VC has run out of credits, the flow control hardware sends a *Stop* signal back to the source to inform it not to forward any more cells on that VC. The flow control hardware then writes the new *i* value to the Shaping Table [14].

The VC on which the cell was transmitted must be added to the Schedule List described in Section 4.1.3:. The *M* value read from the shaping table is added to the value of an internal circular counter to obtain an address into the Schedule Pointer Table described in Section 4.1.2:. The circular counter is used as an index to indicate which cell slot is being serviced. The *Tail Pointer* is read from the Schedule Pointer table and is used to add the VC to the end of the queue in the Schedule List [13].

4.1.5: *Credit Return Hardware*

The credit return hardware is a FPGA that updates the Schedule List and increments the credits of the VCs scheduled for the cell slot indicated by the circular counter. The increment in the *i* value in the shaping table signifies the return of a credit that had been previously scheduled by the flow control hardware [14].

The value of the circular counter is the address of the Head pointer of the VC queue that needs to be serviced. The credit return hardware may have multiple VCs that need to have their credits returned at a given time. Since only one VC can be serviced per cell slot (a hardware limitation), the credit return hardware increments by one the credits of the VC at the head of the queue and then removes the VC from the queue. The second VC in the queue is serviced in the next cell slot, and so on, until the queue is empty. When one queue is emptied, the Controller starts updating the credits of the VCs in the next queue [13]. This kind of scheduling can lead to a skew in the credit return process [14]. Figure 11 shows an example of a possible credit return schedule and Figure 12 shows the actual return of credits as performed by the hardware. The time lost due to the skew in credit returns can be reduced if the aggregate average rate of all VCs on a link is less than the total capacity of the link [15].

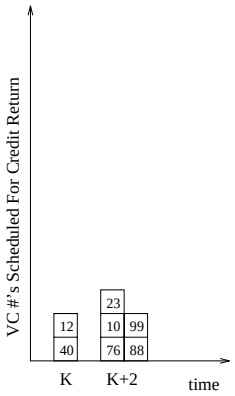

Figure 11. Example of Schedule of Credit Return

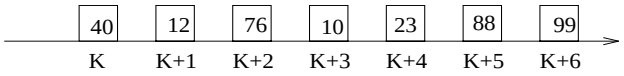

Figure 12. Example of Actual Hardware Credit Return

If the new *i* value becomes larger than a threshold value, meaning that the VC has credits to transmit, the credit return hardware sends a *Start* signal back to the source informing it to forward cells on that VC.

4.1.6: *FPGA Implementation*

The I-out-of-M controller is implemented on a Xilinx XC3195 FPGA using 166 out of 484 Configurable Logic Blocks and 93 out of 176 I/O pins. The interface to the Controller includes: a 16-bit I/O data bus and a 12-bit address bus for the Shaping Table, a 16-bit I/O data bus and a 16-bit address bus shared between the Schedule List and Schedule Pointers Table, and control signals for all the memory structures. The Controller runs at 25 MHz with a cell slot duration of 520 ns. The three memory structures are stored in twelve SRAM chips: four 4k $\times$ 8 SRAMs for the Shaping Table, four 4k $\times$ 8 SRAMs for the Schedule Pointers Table, and four 16k $\times$ 8 SRAMs for the Schedule List. The maximum throughput supported by the Controller is 622 Mb/s, which is SONET OC-12c rate [13, 14].

4.2: Shaping Measurement System

The shaping measurement system can be divided into two functional blocks, the cell count sub-system and the burst count sub-system. The cell count sub-system is a hardware unit that is responsible for counting the number of cells that have been transmitted by a VC. The burst count sub-system determines the number of bursts that have occurred in the measurement interval [17]. The cell count and burst count system use two memory structures to store each of their values. There are two identical banks of these structures that allow the LCP to operate on data in one of the memory structures while the measurement sub-systems write data to the other [14].

4.2.1: Measurement Table

The Measurement Table contains the information needed to calculate the bandwidth utilization of a given VC. Each entry in the table contains a 24 bit unsigned value, which is the number of cells that have gone by on a VC (*cell count*) and a 24 bit unsigned value, which is the number of bursts in which a VC has participated (*burst count*). These are the values that are used to calculate the average rate and average burst described in Section 2.1:. The current size of the measurement table is 128 entries [17].

4.2.2: Translation Table

The Translation Table provides the mapping of VC's to entries in the measurement table. If a VC is not being measured, the mapping entry for that VC needs to be zero. This is to avoid multiple VC entries pointing to the same location in the measurement table. When a VC is removed from the measured list, the entry for that VC is set to zero. The translation table is a 4096 entry table indexed by the VCI [17, 14].

4.2.3: Cell Count Sub-System

The cell count mechanism reads the bus to determine the value of the VC being transmitted. It then uses the translation table to determine if the VC is being measured. If the VC is being measured, the hardware increments the *cell count* field for that VC index in the measurement table [17, 14].

4.2.4: Burst Count Sub-System

The burst count subsystem reads the bus to identify which VC is transmitting. It then uses a time stamp based state machine for that VC to determine the time stamp for the last cell transmitted on that VC (*previous cell time stamp*). It also maintains a time stamp of the last time the line was idle (*idle cell time stamp*). The hardware detects an idle line when two consecutive idle cells are seen. These time stamps are required to determine if the current cell is part of a new burst or a continuing burst. The hardware subtracts the value of *present cell time stamp* from the *previous cell time stamp* for that VC and also calculates the time difference from the *present cell time stamp* and the *idle cell time stamp*. If the first value is greater than the second value then this is the first time the VC is participating in a burst and this means that the *burst count* field for the VC needs to be incremented; otherwise, the VC has already transmitted during this burst so the burst count is not incremented [14]. A special condition might occur when a VC is stopped due to lack of credits. In this case, the *stopped flag* is used in combination with the burst count

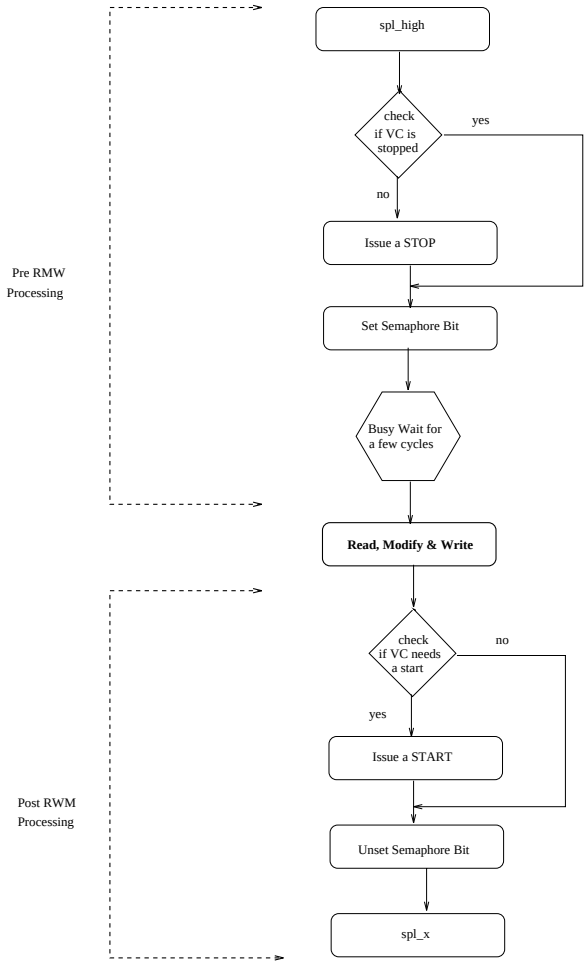

Figure 13. Algorithm used to modify the I and M values in pacing table

as a solution [17]. The *stopped flag* is set when a VC has run out of credits. If the *stopped flag* was set for a particular VC when a burst ended, the number of bursts for that VC is not incremented when a new burst begins. The reasoning for this is, if the VC had credits when the burst ended it would have been able to continue sending the cells in its queue and the second burst would not have occurred. By not incrementing the number of bursts for that VC the same result has occurred [14]. This could possibly lead to a concatenation of every burst in a measurement interval. While this result is not an accurate measure of the average burst size for the source it does not adversely affect the algorithm because the adaptation for I is not based on the actual value of the average burst parameter, but is based on the value of the average burst parameter relative to the previous value for I.

4.3: Hardware - LCP Interaction

The LCP and I-out-of-M controller need to interact at the end of each measurement interval. Once the end of a measurement interval is reached, the measurement banks in the shaping measurement system are switched and the shaping measurement table is read to obtain the number of bursts and the total number of cells sent per VC [15]. The dy-

namic bandwidth allocation process on the LCP uses this information to calculate the average rate and average burst size for each VC [14]. Based on these measurements and the algorithm provided in Section 2:, the LCP determines new *I* and *M* values.

The difference between the old and new *I* and *M* values specifies the number of credits that need to be added or taken away from the current credit balance maintained in the *i* field of the shaping table. The LCP must thus engage in a Read-Modify-Write cycle on the data in the *i* field in the shaping table.

One important aspect of the implementation is that the credit decrements performed by the flow control hardware as ATM cells go by will be stopped during the Read-Modify-Write cycle by stopping the VC. This ensures that no cells come across the crossbar for that VC and therefore no decrement will occur since the flow control hardware will never see a cell on the VC in question [15].

The credit increment will also be disabled during the Read-Modify-Write cycle to avoid losing a returned credit. By setting the semaphore bit in the *m* field of the shaping tables, the credit return hardware is notified that the VC is being updated by the LCP and that the clock needs to be frozen until the semaphore bit is unset [15].

The algorithm used by the adaptive shaping process to implement a safe Read-Modify-Write cycle on the shared data is shown in Figure 13 [14].

The *M* value is never changed by the hardware and therefore the LCP can write the m value via a register write to the desired location in the *M* tables at any time.

5: Conclusion

In this paper we presented the I-out-of-M algorithm, which permits the adaptive control of a source based on indirectly measured parameters. The disadvantages of static control mechanisms provide the motivation for this work. The adaptive I-out-of-M algorithm uses two measurement parameters (*average rate* and *average burst*) for control and provides two parameters (*I* and *M*) that could be directly translated into policer parameters for network control. The simulation results show that this adaptive algorithm does provide source rate and source burst control and smaller delay than static traffic shaping schemes. Finally, a hardware/software implementation of this algorithm is described, proving the efficiency of the algorithm. We conclude that the adaptive I-out-of-M algorithm is a mechanism for real time traffic control desired for information streams generated by the next generation high performance distributed computing applications.

References

[1] D. Anderson, R. Herrtwich, and C. Schaefer. Srp: A resource reservation protocol for guaranteed performance communication. Technical Report TR-90-006, International Computer Sciences Institute, University of California - Berkeley, February 1990.

[2] C. Braun. Master's thesis, University of Kansas, August 1994.

[3] S. Chong, S. Li, and J. Ghosh. Dynamic bandwidth allocation for efficient transoprt of real time vbr video over atm. In *Proc. IEEE INFOCOM*, pages 1c.2.1–1c.2.10, 1994.

[4] I. Cidon, I. Gopal, and R. Guerin. Bandwidth management and congestion control in planet. *IEEE Communications Magazine*, 1991.

[5] T. Faber, L. Landweber, and A. Mukherjee. Dynamic time windows: Packet admission control with feedback. In *Proc. ACM SIGCOMM*, pages 124–135, 1992.

[6] ATM Forum. ATM User-Network Interface Specification, Version 3.0. ATM Forum, June 1993.

[7] Y. Gong and I. Akyildiz. Dynamic traffic control using feedback and traffic prediction in atm networks. In *Proc. IEEE INFOCOM*, pages 1c.3.1–1c.3.7, 1994.

[8] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent capacity and its applications to bandwidth allocations in high speed networks. *IEEE J. Select. Areas Commun.*, 9(7):968–981, September 1991.

[9] L. Gun and T. Tedijanto. Dynamic bandwidth estimation and allocation in high speed networks. Ibm technical report, IBM-RTP, 1993.

[10] H. Lee and J. Mark. Capacity allocation in statistical multiplexing of atm sources. In *Proc. IEEE INFOCOM*, pages 6a.2.1–6a.2.8, 1994.

[11] G. Minden, J. Evans, D. Petr, and V. Frost. An ATM WAN/LAN gateway architecture. In *2nd IEEE Symp. High Perf. Dist. Comp.*, pages 136–143, July 1993.

[12] I. Richer. The MAGIC project. In *Proc. Fourth Gigabit Testbed Workshop*, Reston, Virginia, June 1993.

[13] Srini Seetharam and Hugo Uriona. Desciption of the i-out-of-m controller - Design 1 implemented in Xilinx 3195 FPGA. TISL Technical Report TISL-9770-26, Telecommunication and Information Science Laboratory, University of Kansas, July 1994.

[14] Vinai Sirkay. Real-time implementation of an adaptive shaper for dynamic bandwidth allocation in atm networks. Master's thesis, University of Kansas, December 1994.

[15] Vinai Sirkay, Douglas Niehaus, and Brian Buchanan. Solution to concurrent access of data in *i* and *m* tables by the line card processor and i-out-of-m controller. TISL Technical Report TISL-9770-25, Telecommunication and Information Science Laboratory, University of Kansas, July 1994.

[16] S. Wang. A novel rate-based flow control and buffer management architecture for atm communication networks. In *Proc. IC3N*, pages 54–62, 1993.

[17] Esmael Yousefi. Desciption of the pacing measurement - Design 1 implemented in Xilinx 3195 FPGA. TISL Technical Report TISL-9770-17, Telecommunication and Information Science Laboratory, Univarsity of Kansas, July 1994.