# Implementation of Flexible ABR Flow Control in ATM Networks

Qin Zheng, Randy Osborne, John Howard, Ross Casley, Doug Hahn, Takeo Nakabayashi

TR96-08 December 1996

**Abstract**

The Available Bit Rate (ABR) flow control specification for ATM networks has recently been completed by the ATM Forum. However, the excessive complexity of the specified scheme and the uncertainty about its performance make it costly and risky to implement ABR flow control in hardware. This paper presents a software approach for implementing ABR flow control in source/destination systems to reduce hardware complexity and allow flexibility and upgradability in implementing ABR flow control. The proposed approach is currently being implemented in DART – an ATM network interface control chip under development by Mitsubishi Electric Corporation.

## 1 Introduction

The Available Bit Rate (ABR) is an important service category defined by the ATM Forum to provide efficient support for bursty data traffic in ATM networks [1]. Together with the Constant Bit Rate (CBR) and Variable Bit Rate (VBR) services which are used to support continuous bit stream traffic, ABR is a key element to enable ATM to become a truly integrated network capable of providing services that are currently provided by circuit switched and packet switched networks.

The main objective of the ABR service is to make efficient use of bandwidth left-over by CBR/VBR traffic. To achieve this goal, the amount of ABR traffic that a source system can inject into a network must be controlled according to the current network load condition. Specifically, when a network is lightly loaded, a source should be allowed to send a large amount of data quickly over the network, while as the traffic load increases, a source's transmission rate must be throttled down to avoid network congestion.

One such ABR flow control mechanism has recently been defined by the ATM Forum based on a rate control scheme. To detect network congestion, a source periodically sends Resource Management (RM) cells along with data cells. These RM cells are turned around at their destination. Switches within a network record their congestion information in the passing RM cells. When a source receives return RM cells, it adjusts its transmission rate according to the network congestion information carried in the RM cells. A more detailed description of the ATM Forum's rate-based ABR flow control mechanism is given in Section 2.

Implementation of a rate-based ABR flow control mechanism imposes a great challenge to source/destination systems for two reasons. First, the mechanism as defined by the ATM Forum uses a set of rather complicated rules and a large number of ABR parameters/variables to handle RM cells and adjust transmission rates. This makes it quite expensive to implement ABR flow control fully in hardware. Secondly, the ABR mechanism as defined by the ATM Forum is not well tested. There are known problems that are left for vendors to solve and there have not been enough analysis and simulations showing the performance of the specified mechanism. It is also not clear if currently defined point-to-point ABR flow control framework is suitable for point-to-multipoint flow control. It is thus highly desirable to implement ABR flow control in a flexible and upgradable way such that one can easily adjust the ABR mechanism to get better performance and/or follow possible changes in the ABR specification.

Though it is obvious to use software to provide the required flexibility, it is not clear how to best partition functions between hardware and software and where and how the software processing is performed (e.g., by using an embedded processor core on chip, a local processor on board, or a host processor in an end system). This paper presents an approach for implementing flexible ABR flow control in software that requires minimum hardware support, in contrast to the hardware complexity of an embedded microprocessor core on chip. By doing a careful partitioning of ABR processing functions and using a novel interface between hardware and software, the same minimal hardware can be used to support ABR processing either by a host processor in an end system or a local processor on board. This gives people the freedom of designing a system most suited for their target applications. For example, a very low cost end system can be implemented by having a host processor do the ABR processing with a reasonably low overhead. We have measured the host overhead for such an approach to be less than 15% on a 100 MHz Pentium machine running Windows NT. On the other hand, a fast and powerful local processor can be used for a high-end network interface card or a network switching node implementing virtual source/destination to get a better performance.

Specifically, the approach proposed in this paper uses hardware to implement per cell based

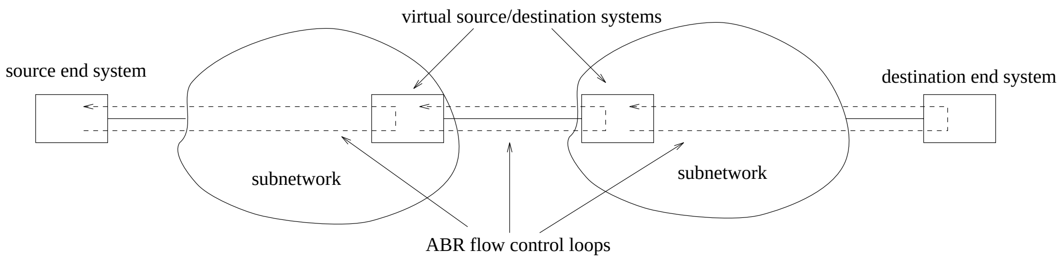

Figure 1: Framework of ABR flow control.

ABR functions and provides an interface to allow an external processor to perform per RM cell based processing in software. The hardware/software interface is designed in such a way that all communication between the hardware and software is realized via access to an external memory, and thus the same interface allows either a host processor or a local processor to perform the ABR processing. Since the per cell based processing, which is time critical and intensive, is performed in hardware, the approach does not degrade ABR flow control performance significantly. Furthermore, this approach incurs only small processing overhead on the external processor, making it feasible to perform such processing on the host processor to realize low cost endstations. At the same time, since important functions such as rate adjustment and RM cell handling are implemented in software, the approach allows a great amount of freedom to modify the ABR functionality via changes in software.

The contribution of this paper is twofold: an elucidation of problem areas in the ATM Forum ABR specification to justify the necessity of flexible ABR flow control and a novel partitioning of functionality between hardware and software to allow low cost, yet very flexible implementations. The paper also describes one such implementation in the DART chip, one of the first generation ATM network interface chips to implement ATM Forum's ABR flow control specification.

This paper is organized as follows. Section 2 gives a detailed description of the ABR flow control scheme as defined by the ATM Forum and discusses its problem areas. The proposed software approach and its implementation in the DART ATM network interface chip are described in Sections 3 and Section 4, respectively. The paper concludes with Section 5.

## 2 Rate-Based ABR Flow Control

We give a description of the ATM Forum's rate-based ABR flow control mechanism and its problem areas in this section. Readers are referred to [1] for a complete specification.

Figure 1 shows a framework under which the ATM Forum's ABR flow control mechanism is specified. A source end system sets up an ABR connection to a destination end system through one or more subnetworks. Virtual source/destination systems can be used to connect subnetworks. An ABR flow control loop is formed between each adjacent pair of source/destination systems by having a source system generating forward RM cells and a destination system turning around received RM cells. Switches within a subnetwork may convey their congestion information to a source system via writing fields in passing RM cells and/or generating RM cells by themselves. The major functions that a source/destination system needs to perform, and their potential problems are described in the following subsections.

## 2.1 RM cell handling

Resource Management (RM) cells are generated by a source system to detect network congestion so it can adjust its transmission rate accordingly. The way in which RM cells are generated has a big impact on the ABR performance. On one hand, generating too few RM cells reduces a source's responsiveness to network load changes, which may cause bandwidth waste and/or buffer overflow at switches. On the other hand, generating too many RM cells increases the overhead of doing ABR flow control. Uncontrolled RM cells may cause network congestion by themselves and reduces network throughput. The ATM Forum adopted a set of rather sophisticated rules for the generation of forward RM cells (i.e., in the direction from a source to a destination):

- Forward RM cells of a connection are generated using the assigned bandwidth to avoid interfering with other connections. This is called in-rate generation of forward RM cells. A connection is allowed to generated out-of-rate RM cells, i.e., with a cell's Cell Loss Priority (CLP) bit set to 1 and without consuming a connection's bandwidth, at a rate no more than 10 cells/section to allow rate ramp-up from zero.
- One RM cell shall be sent for every Nrm cells sent for a connection to keep a proper proportion of RM cells within a cell stream to ensure the responsiveness and limit the overhead of ABR flow control. Nrm is an ABR parameter with a default value of 32.
- For connections sending cells at extremely low rates, one in-rate forward RM cell can be sent in every Trm seconds, where Trm is an ABR parameter with a default value of 100 milliseconds.
- To avoid forward RM cells blocking data cells and backward RM cells, no more than one forward RM cell shall be sent in every two cells sent in the presence of data or backward RM cells.

The above rules try to strike a compromise between responsiveness and overhead of ABR flow control. But there are still problems. For example, a connection sending at a low rate (possibly due to a previous network congestion) may have to wait for 100 milliseconds to send a forward RM cell, and then wait for the return of the cell to get a chance to ramp-up its rate. This response speed may not be satisfactory for many LAN applications. On the other hand, a large number of connections sending RM cells at a 10 cells/second rate may also cause a problem to a network.

Just like the generation of forward RM cells at a source, a destination must also be very careful in turning around RM cells. The ATM Forum uses the following rules for sending backward RM cells (i.e., in the direction from a destination to a source).

- RM cells shall be turned around using the bandwidth assigned to the backward direction of the connection. Only if a second RM cell is received before the first RM cell is turned around, can the first RM cell be sent back to its source out-of-rate, i.e., with it CLP bit set to 1 and not using the backward bandwidth. It is also allowed for the first RM cell to be discarded.
- In case there is a conflict with forward RM cells or data cells, a forward RM cell has a transmission priority over a backward RM cell. After the transmission of a forward RM cell, the first backward RM cell has a priority over data cells, and then data cells will have

priority over backward RM cells until the generation of another forward RM cell. This dynamic priority setting scheme avoids possible bandwidth starvation problem for either backward RM cells or data cells.

One problem with the above RM cell turn around scheme is the case where forward and backward bandwidths of a connection are not well balanced (i.e., backward bandwidth $< 1/Nrm$ of the forward bandwidth). In this case, a large number of RM cells are either sent back with CLP=1 (i.e., they are dropped first by a network in case of congestion) or discarded by a destination. Loss of backward RM cells can significantly degrade the ABR performance. It should be noted that unbalanced bandwidth allocation is not unusual at all. To make efficient use of network resources, no significant bandwidth should be allocated to one direction of a connection if most of its traffic is flowing in the other direction. Also for point-to-multipoint connections, the backward bandwidth is always zero, thus all RM cells are either turned around out-of-rate or dropped at the destination. For this reason, it is not clear if the point-to-point ABR flow control framework as defined today can also accommodate future point-to-multipoint flow control schemes.

### 2.2 Rate control and adjustment

The transmission rate of an ABR connection is controlled to be under an Available Cell Rate, ACR. ACR is initially set to an Initial Cell Rate, ICR, and then adjusted in a range between a Peak Cell Rate, PCR, and a Minimum Cell Rate, MCR, in a way described below. The ATM Forum's rate-based scheme assumes a source/destination system to have a rate controller (or traffic shaper) capable of controlling dynamically changing rate accurately for each individual connection. The impact of inaccurate rate control (e.g., the widely used rate group approach implementing 8 - 16 rates) to the ABR performance is not yet clear.

The way in which ACR is adjusted has a direct impact to the ABR performance. Ideally, ACR should be increased to PCR when a network is not congested to make the best use of available bandwidth, and ACR should be reduced to MCR in case of congestion to avoid buffer overflows in a network. However, due to delays between the occurrence of congestion and the time when a source is notified by a backward RM cell, such dramatic rate changes may generate large oscillations of ACR. In addition, a source needs to have a self-limiting property, meaning that it reduces its rate when not receiving backward RM cells for a long period of time. To achieve these objectives, the ATM Forum defined the following rules for rate adjustment:

- When a backward RM cell is received with its CI (congestion indication) bit set to one, the ACR shall be reduced exponentially in reacting to network congestion, and if the backward RM cell has both CI and NI (no increase) bit set to zero, the ACR may be increased additively (to a rate no more than PCR).

  One problem with this type of binary rate adjustment is the slow response to network load changes. It may take a while for a source to settle down on an ideal ACR, causing either underutilization of network bandwidth or large queue build-up at congested switches. Another well-known problem is the unfairness in bandwidth allocation among ABR connections caused by a "beat-down" phenomenon. Specifically, a connection which goes through a large number of switches is more likely to receive an backward RM cell with the CI bit set than connections over a smaller number of switches, thus it tends to have an ACR smaller than that of other connections.

- To improve a source's responsiveness to network load changes and deal with the "beat-down" problem, an ER (explicit rate) field is included in RM cells which can be used by a switch or a destination system to directly set the transmission rate of a source to a desired value. When a source receives a backward RM cell, it sets its ACR to the minimum of that calculated from the CI/NI bits and the value contained in the ER field (and no less than MCR as always). The explicit rate approach is expected to improve a source's responsiveness and solve the beat-down problem as long as switches have the ability of setting the ER fields correctly, which by itself is a challenging problem that has been left to switch vendors to deal with.

- Besides reacting to feedback from a network, a source must also adjust its transmission rate occasionally when it is not receiving feedback. This self-limiting feature is implemented with following two rules:

  - a connection which has been idle for a certain period of time, measured as the time interval between sending two consecutive forward RM cells exceeding a pre-specified value (0.5 second by default), should reduce its ACR to ICR to protect a network from a situation where a large number of idle connections burst at high rates simultaneously, and
  - a connection should reduce its ACR exponentially when it believes that its feedback loop is broken or experiencing extremely long delays (i.e., when the number of outstanding RM cells reaches a pre-specified threshold).

  The ATM Forum also provides a number of "use-it-or-lose-it" rules to reduce a connection's ACR to a value which approximates the actual cell transmission rate. This is to protect a network from being congested by simultaneous bursts of a large number of connections previously sending at low rates but retaining large values of ACR. However, a side effect of using a "use-it-or-lose-it" rule is that it may severely hinder the performance of request-response type applications. Specifically, applications sending short bursts of data may not be able to get enough bandwidth even if the network is lightly loaded. Due to this, the ATM Forum decided to leave it to vendors' latitude to decide whether or not to implement one of the recommended "use-it-or-lose-it" rules (or implement a rule of their own).

In summary, the ATM Forum has carefully specified a set of rules for source/destination systems to follow. These rules are expected to work well in most situations. However, it is difficult to predict real world performance due to many open issues as discussed above and a lack of theoretical analysis and simulations of the specified scheme. Some fixes and/or enhancements to the scheme might be needed as people gain more experience with ABR flow control. Support for multicast ABR flow control may also introduce some adjustments to the current ABR flow control framework. It is thus very important for vendors to be prepared for these changes when implementing their ATM products with ABR flow control.

## 3 Software ABR Flow Control

There is a spectrum of ways to partition the functionality of ABR flow control between hardware and software to maintain flexibility. This section describes a particular partitioning and interfacing between hardware and software that maintains flexibility and performance but requires

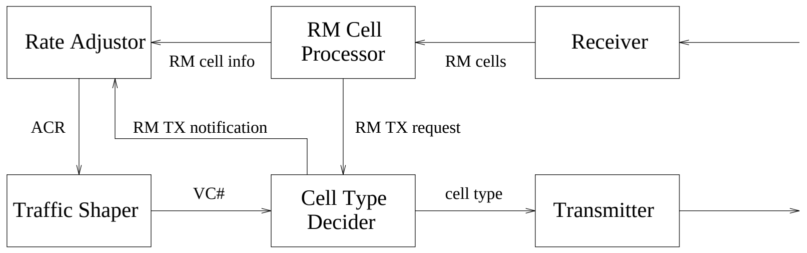

Figure 2: ABR flow control functional blocks.

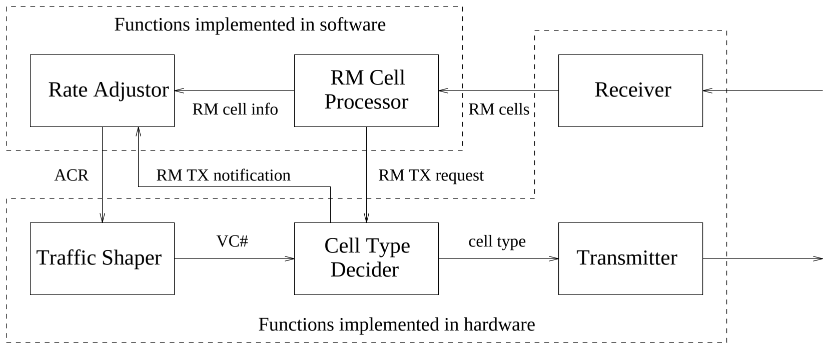

Figure 3: Partition of ABR flow control functions.

minimum hardware support. The approach allows a very low cost end system implementation by shifting most of the ABR processing to a host processor. It also allows a dedicated processor for high performance without requiring additional hardware on chip. The proposed approach can be used either in an end system (i.e., a network interface card) or in a switching node implementing a virtual source/destination as shown in Figure 1.

Figure 2 shows functional blocks required to implement ABR flow control and interfaces between the blocks. On the transmission side, a traffic shaper assigns transmission slots to connections in such a way that their transmission rates are controlled under the specified values (i.e., ACR). When a connection is selected for sending a cell, a Cell Type Decider decides whether to send a data cell, a forward RM cell, or a backward RM cell, according to the RM cell handling rules described in Section 2.1. In certain cases, e.g., on sending a forward RM cell, the Rate Adjustor is notified to perform possible rate adjustments. The transmitter then assembles a cell and sends it out to the network. On the receiving side, received RM cells are forwarded to an RM Cell Processor. For a forward RM cell, the RM Cell Processor modifies the cell (e.g., the direction and CI bits), stores it temporarily, and notifies the Cell Type Decider to send the cell back to its source at an appropriate time and in an appropriate way. On receiving a backward RM cell, the RM Cell Processor retrieves feedback information from the cell and forwards the information to a Rate Adjustor which calculates a new ACR for the connection.

With the ABR functional blocks identified above, we decided to use a partition as shown in Figure 3. With this partition, functions performed on a per-cell basis are implemented in hardware and those performed on an per-RM-cell basis are implemented in software. It also implements functions that require flexibility the most, e.g., rate adjustment and RM cell handling, in software.

Figure 4 shows an implementation of the above functional partition, where the software part of ABR processing is performed by an external processor and the interface between the processor and chip hardware is implemented by a Control Table and a Processor Requester

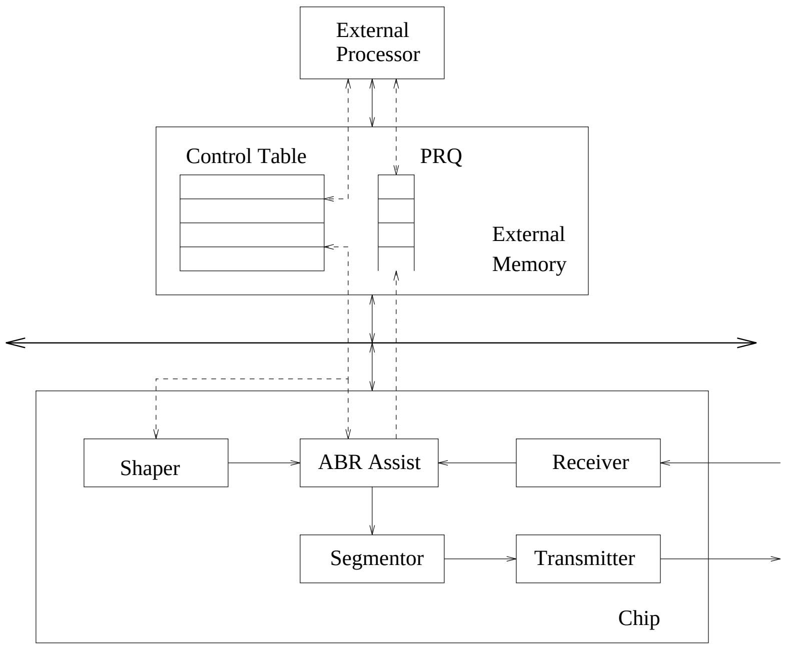

Figure 4: Source/destination system implementation.

Queue (PRQ) residing in an external memory, and a hardware ABR Assist on chip. Detailed operations are described in the following subsections.

## 3.1 ABR request submission

Software ABR processing is initiated by the ABR assist's submission of an ABR request to the PRQ. The processor either checks PRQ periodically and/or reads PRQ upon receiving an interrupt from the ABR assist. There are four types of ABR requests:

- FRM_TX: a request of type FRM_TX is submitted to the PRQ when a forward RM cell is sent. A FRM_TX request contains a time stamp recording the time when the forward RM cell is sent. Its main purpose is to implement the self-limiting feature as specified by the ATM Forum. Specifically, when a processor finds that the difference between time stamps of two consecutive RM cells is larger than a pre-specified value, it reduces ACR to ICR.

- FRM_RX: a request of type FRM_RX is submitted to the PRQ when a forward RM cell is received. A FRM_RX request contains the received RM cell and a bit indicating whether or not a previously received forward RM cell has been turned around to help the processor decide whether to turn around this RM cell in-rate or out-of-rate.

- BRM_RX: a request of type BRM_RX is submitted to the PRQ when a backward RM cell is received. A BRM_RX request contains the received RM cell and a *tx_cell_count* field recording the total number of cells transmitted for the connection. This field is used by the processor to set a limit to the total number of cells that a connection can send before it is stopped.

- OUT_LIMIT: a request of type OUT_LIMIT is submitted to the PRQ when the total number of cells that a connection has sent, *tx_cell_count*, reaches a limit *tx_cell_limit* set by the processor. An OUT_LIMIT request contains the current value of *tx_cell_count*. After submission of the request for a connection, the transmission of the connection is stopped. This mechanism is used to self-limit the transmission of a connection when a processor fails to timely process requests in the PRQ. It can also be used to implement credit-based flow control schemes such as the one specified in [2].

In addition for ABR flow control, PRQ is also a convenient place for submitting non-ABR requests to a processor (e.g., error handling and OAM cell processing).

## 3.2 Handling of RM cells

On each transmission opportunity, a hardware ABR assist determines the type of cell that a connection should send by reading state and control fields in the control table. An external processor can thus control the generation and turn around of RM cells by writing the control fields in the control table. Detailed operations are described below.

- Generation of forward RM cells.
  A hardware ABR assist controls the generation of proportional RM cells. Specifically, it maintains a *cell_since_frm* field in the control table for each connection recording the number of in-rate cells sent since last in-rate forward RM cell. A forward RM cell is sent when *cell_since_frm* reaches Nrm.

  Generation of non-proportional in-rate forward RM cells, e.g., those generated when ACR is very low, is controlled by the external processor. The control table has two bits for this purpose: *frm_in* and *frm_out*. To send an RM cell, the external processor sets the *frm_in* bit to be different from *frm_out*. The hardware ABR assist sends a forward RM cell at the next transmission opportunity when it detects this condition. After transmission of an RM cell, the hardware ABR assist sets the *frm_out* bit to make it equal to *frm_in*. The reason for using two control bits instead of a single bit is to avoid possible race conditions in updating the same bit by the external processor and the ABR assist.

  Out-of-rate RM cells, e.g., those generated when ACR of a connection is zero, can be generated by having the external processor inject them into a network as raw cells.

  Since the frequency of proportional RM cell generation is controlled by the Nrm field in the control table and the generation of non-proportional forward RM cells is entirely controlled by an external processor, a wide range of forward RM cell generation schemes can be implemented without requiring changes to the hardware.

- Turn around of backward RM cells.
  After processing a received forward RM cell, the external processor turns it around back to its source as a backward RM cell. This is implemented by using a pair of control bits *brm_in* and *brm_out* in the control table in a way similar to that used for the generation of non-proportional in-rate forward RM cells. Specifically, to turn around an RM cell, an external processor sets the *brm_in* bit to be different from *brm_out*, and after the transmission of the backward RM cell, the hardware ABR assist sets the *brm_out* bit to make it equal to *brm_in*.

  Besides $brm\_in \neq brm\_out$, a hardware ABR assist also checks the following conditions in making a decision on sending a backward RM cell.

  1. Since the forward RM cells have a priority over backward RM cells, no backward RM cell is sent if the conditions for sending a forward RM cell is satisfied.

2. A control bit, $frm\_after\_brm$, is used to determine if a backward RM cell should be sent in the presence of data cells. The bit is set after the transmission of an in-rate forward RM cell and reset after the transmission of a backward RM cell. It is used to implement the dynamic backward RM cell transmission priority as described in Section 2.1. Specifically, if $frm\_after\_brm = 1$, a backward RM cell is sent. Otherwise, a backward RM cell is sent only in the absence of data cells.
3. To give an external processor some control over the transmission priority of a backward RM cell, a *brm_pri* bit is used in the control table to allow an external processor to assign a backward RM cell a higher priority over data cells. The hardware ABR assist resets the *brm_pri* bit after the transmission of the backward RM cell.

Like sending out-of-rate forward RM cells, backward RM cells can be turned around out-of-rate by an external processor by injecting them into a network as raw cells.

### 3.3 Rate adjustments

An external processor changes the transmission rate of a connection by writing a rate field in the control table. The rate field, storing a value of 1/ACR, is read by a hardware traffic shaper to schedule the transmission time for a next cell after the transmission of the current cell. There are three cases when an external processor may need to change the transmission rate for an ABR connection.

- On receiving a BRM_RX request, the processor reads the received backward RM cell from the PRQ, adjusts ACR according to rules specified by the ATM Forum as described in Section 2.2.
- On receiving a FRM_TX request, the processor checks if a connection should self-limit itself as described in Section 2.2.
- On receiving an OUT_LIMIT request, the processor marks the connection as in a STOPPED state. The connection should be re-activated whenever its total number of cells sent falls below a cell transmission limit set for the connection.

To summarize, the above described software scheme realizes a very flexible ABR flow control implementation at source/destination systems. Changes in the rate adjustment algorithm, RM cell handling rules and RM cell formats can be easily accommodated via software changes only. It also allows a very flexible configuration of a source/destination system. One may use a dedicated processor for fast ABR processing or minimize the cost of the ABR processing by sharing the processor with other applications, such as using a host processor at a source/destination end system. This later configuration is implemented in Mitsubishi Electric's DART ATM network interface control chip and is described in the next section.

## 4 ABR Flow Control in DART Chip

DART is an ATM network interface control chip currently under development by the Mitsubishi Electric Corporation. It integrates ATM adaptation layer processing (AAL5 and AAL0) and a PCI bus interface into a single chip for full duplex 155Mbps transmission. The design incorporates advanced features to support high performance (high bandwidth and low latency applications) and flexible flow control via software. The device meets the latest ATM standards. Its major features include

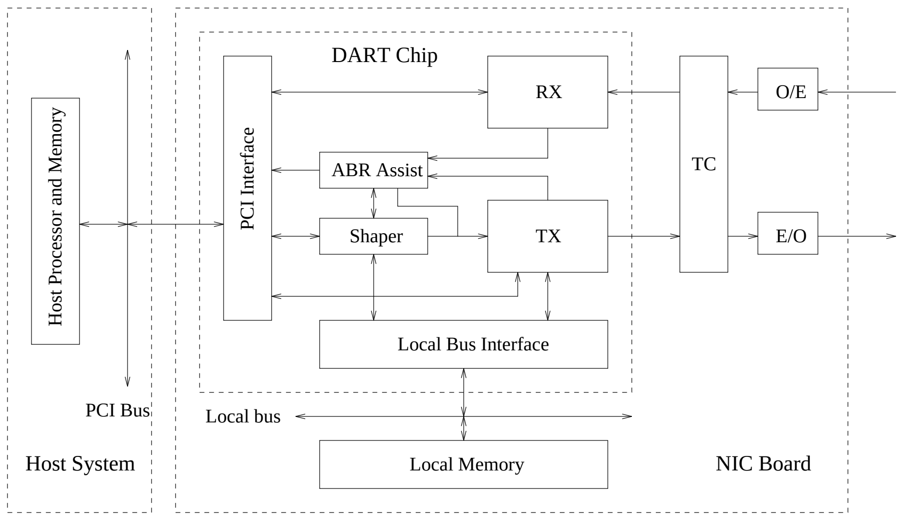

Figure 5: DART chip and system configuration.

1. First commercially available NIC chip allowing direct, protected access from application to network, bypassing OS, to
   - eliminate OS overhead in data transfer for high bandwidth
   - support small messages and low latency communication
   - allows application-specific protocols
2. Software ABR flow control supports ATM flow control standards such as the ATM Forum rate-based scheme and the Credit Consortium credit-based scheme with flexibility to follow standard evolution and implement proprietary schemes.
3. Full-featured transmit unit performing per-connection traffic shaping capable of any transmission rate for up to 32K connections. Four transmission priority levels supporting multiple traffic classes (CBR, rt-VBR, nrt-VBR, ABR, UBR).
4. Optional processing of messages between the network and application intended for low level processing of AAL0 and AAL5 frames. Possible uses include remote operations for low latency communication (e.g. read and write of memory in other computers), cyclic communication in real-time systems, filtering of unwanted messages, and fast/cheap demultiplexing.

Figure 5 shows the internal structure of the DART chip and one possible end system configuration. We will describe the traffic shaper, ABR assist and host ABR processing in this section. Readers are referred to [3, 4] for details of other functional blocks.

## 4.1 Traffic Shaper

To support accurate rate control for a large number of connections with a wide range of rate values, the DART chip uses an enhanced timing chain approach for traffic shaping as shown in Figure 6. Each component of the shaper is explained below.

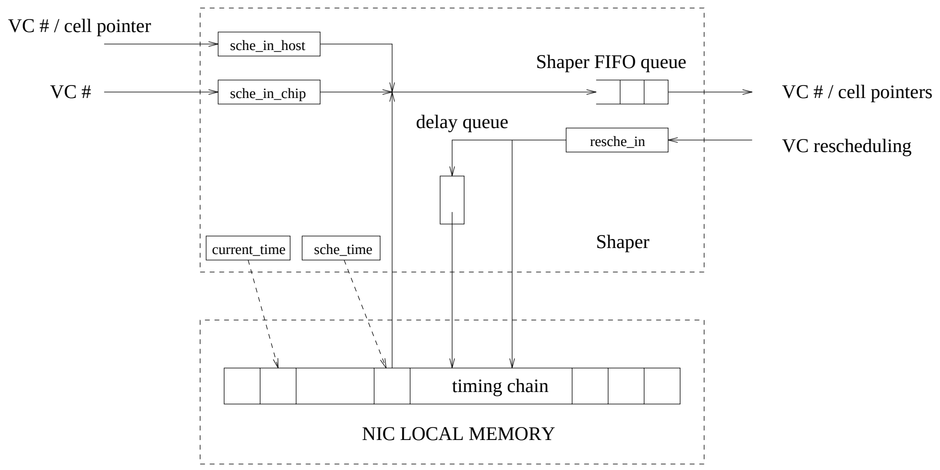

Figure 6: Traffic shaper.

- Timing chain.
  A timing chain is an array of slots residing in a local memory. Each slot is a 32-bit entry representing one cell transmission time over the output link. Each timing chain slot can store one connection ID or a linked list of connection IDs with links stored in the control table. A current_time pointer on the chip points to a slot representing the current time and moves forward one slot per cell time. A sche_time pointer points to a slot from where connection IDs are being dequeued from the timing chain. The sche_time pointer moves forward one slot after it dequeues all connection IDs in a slot until it catches up with the current_time pointer. To rate control a connection with a cell rate R, after the transmission of a cell, the connection ID is re-scheduled into a slot which is 1/R slots away from the slot pointed to by the current_time pointer.

- Shaper FIFO queue.
  The shaper FIFO queue is used to store connection IDs dequeued from the timing chain or pointers of raw cells to be transmitted. To implement multiple transmission priorities, the shaper FIFO is composed of four priority FIFO queues. Connection IDs or cell pointers are queued at a priority FIFO queue corresponding to their assigned priority levels. Items stored in priority FIFO queues are read out by an ABR assist in the order of their priority levels.

- resche_in register.
  A resche_in register is used to reschedule a connection into the timing chain after the transmission of a cell. The resche_in register holds a connection ID and a value *dt* indicating that the connection should be scheduled *dt* slots away from the current_time pointer in the timing chain. Different type of traffic shaping algorithms, e.g., leaky bucket traffic shaping for VBR connections, can be implement by using different ways of calculating *dt*.

- Delay queue.
  Due to the limited length of a timing chain, a connection may not be able to be scheduled in the timing chain if *dt* is too large. A delay queue is used to store such connection IDs. The delay queue is implemented as a linked list of connection IDs. The head of the linked list is dequeued periodically to see if it can be scheduled into the timing chain. If it still does not fit, it is queued back at the end of the linked list. This mechanism makes the

shaper capable of supporting a connection with an arbitrary low transmission rate.

- sche_in_chip and sche_in_host.
  The sche_in_chip register is used by the chip to initiate scheduling of a connection (i.e., when the connection is not already scheduled in the timing chain) by writing the connection ID into the register. Items written into the sche_in_chip register are directly moved to the shaper FIFO queue.

  The sche_in_host register is used by the host to initiate scheduling of a connection. It also accepts a cell pointer. It provides a means for a host to re-activate a stopped connection and send out-of-rate raw cells.

## 4.2 ABR assist and host ABR processing

The DART chip allows a low-cost end system implementation by using a host processor for the ABR processing. With this configuration, the control table is placed in a local memory on board for easy access by the chip through a local bus. The PRQ resides in host memory for easy access by a host processor. Since the chip only writes to PRQ, large PCI latency is avoided by a posted write mechanism.

An ABR assist implements the hardware part of the ABR processing. It performs the following major functions.

- Dequeue an item from the shaper FIFO when the transmitter is ready to send a cell, and re-schedule a connection in the shaper after transmission of an in-rate cell for the connection. Notice that this function is performed for all types of connections.
- Determine a cell type for a connection by reading control table in a way described in Section 3.2.
- Submit ABR requests to the PRQ as described in Section 3.1.
- Interrupt the host processor in a way described below.

To reduce a host processor's ABR processing overhead, the ABR assist implements a special interrupt mechanism which enforces a minimum interrupt period. Specifically, instead of generating one interrupt for each request submitted to the PRQ, the chip generates just one interrupt for all requests submitted within a user specified interrupt period. Fewer number of interrupts and batch processing of ABR requests reduces the overhead incurred by a host processor. Since ABR processing is normally done on a per-RM-cell basis and not time critical, this mechanism is a useful means to allow users to reach a compromise between responsiveness and software overhead of ABR flow control.

Upon receiving an ABR interrupt from the chip, the host processor dequeues and processes requests in the PRQ one by one. Processing required for each type of request is described in Section 3.3. It should be noted that operations described in Section 3.3 implement ATM Forum's ABR flow control specification only. Additional/alternative processing can be performed to enhance ATM Forum's flow control scheme or implement other flow control schemes. The DART chip also has additional hardware support to allow implementation of Quantum Flow Control as specified by the Flow Control Consortium [2]. Space limitations preclude a discussion of further details here.

### 4.3 Host ABR processing overhead

One important issue that needs to be addressed for software ABR processing using a host processor is how one can best compromise between the overhead incurred by the host processor and the performance of the implemented ABR flow control. On one hand, the host should be interrupted as infrequently as possible to minimize the overhead. On the other hand, ABR requests submitted to the PRQ should be processed as soon as possible to improve the responsiveness of ABR flow control. In the rest of this section, we present some overhead measurement that have been performed and discuss the impact of processing delay to the responsiveness of ABR flow control.

Measurements on a 100MHz Pentium system indicate 5$\mu$sec to enter and exit the Windows NT low level driver[1] and about 7$\mu$sec on average to process three ABR requests (FRM_TX, FRM_RX, BRM_RX). At full line rate of 155 Mbps, this overhead is experienced on average every 86$\mu$sec (32 cell times). In other words, if the minimum interrupt period is set to one RM period (i.e., 32 cell times), in the worst case it takes no more than 15% of a host's processing power to support full line rate ABR transmission and receiving simultaneously. In a normal case, we expect less overhead for the following reasons:

- When an end system is sending and/or receiving at high rates, a network driver is involved a lot in transmitting and receiving data frames. A driver can be programmed to also check the PRQ after submitting or receiving a frame to/from the network interface card so that no ABR interrupts need to be generated when the host is performing driver tasks. This can reduce the ABR interrupt overhead and also improve the responsiveness of software ABR flow control.

- Host ABR processing overhead occurs only when transmitting or receiving ABR traffic. All other traffic types, i.e., CBR/VBR/UBR, do not require a host's involvement for flow control. So the host overhead is significantly smaller than 15% in the presence of other types of traffic and/or when it is not simultaneously sending and receiving ABR traffic at full line rate.

- Since the processing of ABR requests is not time critical, the host can batch ABR requests to amortize overhead by increasing the minimum interrupt period. A longer interrupt period is equivalent to making the feedback loop longer. We will investigate its impact to the flow control responsiveness and performance by using an ATM network simulator developed by NIST [6] and a VERILOG model developed for the DART chip.

In any case, the rapid roll out of fast PCs will diminish the relative overhead (We plan to redo the measurements on 160MHz Pentium-Pro PCs). The configuration of an end system with a host processor doing ABR processing provides a low-cost solution for the first generation ATM products implementing ABR flow control. In the case that the responsiveness of ABR flow control and host overhead requirements can not be met, a dedicated local processor on board can always be used with the proposed software approach for a high-end network interface card.

## 5 Conclusions

In this paper, we discussed the necessity of implementing flexible ABR flow control in ATM networks and presented an approach for doing ABR flow control in software. With a careful

[1]Kindly performed for us by Brad Chen and Yasuhiro Endo at Harvard University [5].

partition of ABR functions and a novel interface between software and hardware, our approach requires a minimum amount of hardware support as compared to implementing ABR fully in hardware or using an on-chip processor core for ABR processing. A very low cost endstation can be realized by having a host CPU do the ABR processing with a reasonably low overhead. It also allows the same minimum hardware to support ABR flow control by a dedicated processor to achieve high performance at a high-end network interface card or in a network switching node. The proposed approach is currently being implemented in a Mitsubishi Electric's ATM network interface control chip.

## Acknowledgement

The authors would like to thank Dr. Hugh Lauer and Dr. Abhijit Ghosh of Mitsubishi Electric Research Laboratories for their management of the DART chip development project.

## References

[1] S. S. Sathaye, "Traffic management specification, version 4.0," *The ATM Forum Technical Committee*, February 1996.

[2] M. Gaddis and W. Kelt, "Quantum flow control, version 2.0," *The Flow Control Consortium*, July 1995.

[3] R. Casley, P. Chai, A. Ghosh, D. Hahn, H. Lam, R. Osborne, and Q. Zheng, "External specification of phase-2 AAL/BIF LSI, version 1.0," *Mitsubishi Electri Research Labs.*, March 1996.

[4] R. Osborne, Q. Zheng, J. Howard, R. Casley, D. Hahn, and T. Nakabayashi, "DART – a low overhead ATM network interface chip," *Mitsubishi Electri Research Labs.*, March 1996.

[5] J. B. Chen, Y. Endo, K. Chan, D. mazieres, A. Dias, M. Seltzer, and M. D. Smith, "The measured performance of personal computer operating systems," *The Proceedings of the 15th ACM Symposium on Operating System Principles*, December 1995.

[6] N. Golmie, A. Koenig, and D. Su, "The NIST ATM network simulator – programmer's guide," *National Institute of Standards and Technology*, December 1994.