CONF-961125--2

SAND--96-8548C

Fast Packet Switching Algorithms for Dynamic Resource Control over ATM Networks[1]

R.P. Tsang

Sandia National Laboratories
Distributed Information Technologies Center
Infrastructure and Networking Research
Livermore, California 94551-0969 USA

P. Keattithananant, T. Chang, J. Hsieh, David Du

Distributed Multimedia Center
Computer Science Department
University of Minnesota
Minneapolis, MN 55455 USA

Abstract

Real-time continuous media traffic, such as digital video and audio, is expected to comprise a large percentage of the network load on future high speed packet switch networks such as ATM. A major feature which distinguishes high speed networks from traditional slower speed networks is the large amount of data the network must process very quickly. For efficient network usage, traffic control mechanisms are essential. Currently, most mechanisms for traffic control (such as flow control) have centered on the support of Available Bit Rate (ABR), i.e., non real-time, traffic. With regard to ATM, for ABR traffic, two major types of schemes which have been proposed are rate-control and credit-control schemes. Neither of these schemes are directly applicable to Real-time Variable Bit Rate (VBR) traffic such as continuous media traffic. Traffic control for continuous media traffic is an inherently difficult problem due to the time-sensitive nature of the traffic and its unpredictable burstiness. In this study, we present a scheme which controls traffic by dynamically allocating/deallocating resources among competing VCs based upon their real-time requirements. This scheme incorporates a form of rate-control, real-time burst-level scheduling and link-link flow control. We show analytically potential performance improvements of our rate-control scheme and present a scheme for buffer dimensioning. We also present simulation results of our schemes and discuss the tradeoffs inherent in maintaining high network utilization and statistically guaranteeing many users' Quality of Service.

Keywords: Asynchronous Transfer Mode (ATM), dynamic traffic control, resource management, rate control, Real-time Variable Bit Rate traffic, continuous media traffic.

[1] This work is supported in part by the Advanced Research Projects Agency (ARPA) through AFB, contract number F19628-94-C-0044, U S WEST Communications, and the Distributed Multimedia Center (DMC), sponsored by US WEST, Honeywell, IVI Publishing, Computing Devices International and Network Systems Corporation.

1 Introduction

Video conferencing, collaborative systems, distance learning, and VOD (video on demand) are all new applications which are based upon the efficient transmission of real-time variable bit rate traffic such as digital video and audio. It is expected that these real-time traffic types will be transported on a fast packet-switch network platform such as the Asynchronous Transfer Mode (ATM).

The ATM standard [3, 10, 13] defines a fast packet switched network where data is fragmented into fixed-size 53 byte cells. It defines the manner in which cells are switched and routed through network packet switches and links. The ATM standard is expected to serve as the transport mode for a wide spectrum of traffic types with varying performance requirements. Using the statistical sharing of network resources, it is expected to efficiently enable multiple transport rates from multiple users with stringent requirements on loss, end-to-end delay, and cell-interarrival delay.

Network resources include processing buffers and link capacity. Traffic control and congestion control policies enforce their objectives through the management of network resources. The objective of traffic control policies is to maintain the Quality of Service (QoS) requirements of traffic flows, i.e., Virtual Circuits (VCs), as well as to avoid a state of congestion. The objective of congestion control policies is to reduce the severity, duration and spread of congestion.. These policies provide resource control by embedding controls into the network elements. An example is a scheduling algorithm at a switch output port which manages the link capacity resource by deciding which cells should be forwarded. Some policies may also rely on special indicators embedded in the traffic itself which are reacted upon by the embedded network controls. An example is a special cell, say a Resource Management cell, which is sent by a congested node to its upstream nodes to trigger a reduction of rate in order to prevent excessive cell loss at a congested buffer.

There are five defined ATM layer service categories: Constant Bit Rate, Non Real-time Variable Bit Rate, Real-time Variable Bit Rate, Available Bit Rate and Unspecified Bit Rate [8]. In our study, we consider Real-time Variable Bit Rate (VBR) traffic. In particular, we consider real-time *bursty periodic* traffic. Continuous media traffic, the predominant form of multimedia traffic, such as video and audio is a bursty periodic traffic source. A bursty periodic stream is distinguished by the appearance of variable size bursts every fixed interval period. For instance, digital coded video consists of a series of burst (frames) where each frame occurs every 30 milliseconds (NTSC digital video standard).

Currently, most proposed traffic control policies have focused on the support of Available Bit Rate (ABR) traffic [2, 4, 8, 11]. ABR traffic is typical computer data traffic which consists of file transfers, email, etc. ABR traffic is distinguished by being non real-time and loss-sensitive. The overall goal of these policies have been high network utilization, although low delay and low loss ratios are also sought.

Currently proposed traffic control policies fall into two categories: rate control and credit-based policies. For real-time traffic, rate control can only be used as long as it doesn't violate the traffic's real-time constraints. Credit type schemes are not suitable for real-time traffic unless some notion of real-time delivery constraints are incorporated. Traffic control for Real-time VBR traffic is considered a difficult area due to the time-sensitive nature of the traffic. So it

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

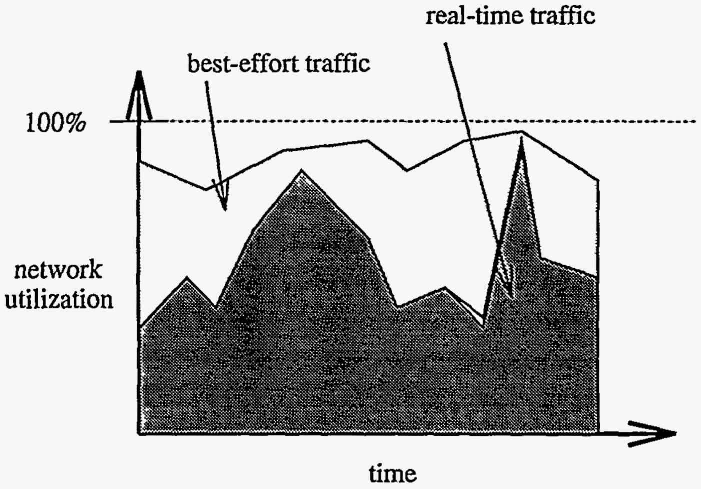

Figure 1: Best effort traffic + real-time traffic

has been proposed [12] in networks with real-time VBR (guaranteed) traffic to allocate peak resources while, for the sake of network utilization, allowing ABR traffic to consume the leftover resources after the guaranteed traffic has been served. The goal is high network utilization (see Figure 1). However, what if the network is primarily used to transport real-time VBR traffic such as video traffic in a video server environment ? Allocating peak resources would be enormously wasteful. Allocating less than peak implies greater network utilization. However then the possibility for unpredictable statistical fluctuations in the traffic arises, and hence cell losses and delays which result in the subsequent degradation of Quality of Service (QoS). Another potential dilemma is that higher network speeds will give rise to much larger quantities of data a network must support, thus contributing to producing large traffic fluctuations. Obviously, some form of traffic control is a necessity for the efficient usage and control of network resources. Flow control policies such as rate control, link-to-link and end-to-end flow control procedures must be developed for real-time VBR traffic.

We propose traffic control (both scheduling and flow control) specifically for continuous media traffic. Since in this type of traffic, data is aggregated into *bursts* of cells, overall performance is more accurately reflected by burst level performance than cell level performance. We present a novel approach to real-time scheduling which operates on the burst level. For real-time traffic scheduling, traffic must be given some notion of priority based upon deadlines so an attempt is made to deliver the 'earliest-deadline' traffic first. Also, the computation of the selection of the highest priority traffic must also be very fast due to the high speeds of these types of digital networks. We incorporate both of these notions in this study.

Recently the notion of Resource Management (RM) cells to enhance the functionality of ATM at the network layer has been proposed by [4, 9, 15, 14]. There are two major ways which have been proposed for using RM cells; one is as a resource reservation technique [4], another is to indicate a changing network condition such as the onset of congestion [9, 15, 14] In [4], the source would initially send an RM cell downstream to the destination. If an intermediate switch cannot accept the request, it drops the RM cell and the source times out. Otherwise, if the destination receives the RM cell, it returns it back to the source. The source then transmits. An "immediate transmission" mode was also proposed where the burst immediately follows the RM cell. If any intermediate switch cannot accept the request, it drops the RM cell and the burst and sends an indicator cell back to the source. None of these methods are appropiate for real-time traffic. In this study, we will explore the concept of dynamic resource reservation using RM cells for continuous media traffic.

In the next section, we describe the overall problem. Section 3 presents the rate control, buffer control and congestion control algorithms. Sections 4 and 5 provide the analysis and simulation results. Section 6 provides the Conclusion.

2 Description of Problem

In this section, we formulate the problem and discuss issues surrounding its formulation and solutions.

Assumptions. In our study, we assume the following.

- All traffic is bursty periodic traffic; a traffic stream consists of variable size bursts occurring every fixed time interval. Obvious examples of such traffic is continuous media traffic such as digital video and audio. For instance, digital video consists of frames (or bursts) of data where every frame corresponds to an image. In order for a viewer to observe jitter-less video, each image must be delivered within a fixed time interval of at least 40 millisec.
- All switches are output buffered. We assume output buffered switches since output buffering is the common denominator type of buffering mechanism found in most ATM switches.
- A *time slot* corresponds to the time it takes to send one cell. For instance, given an OC-3 155 Megabits/sec link, each time slot is approximately 2.75 microseconds. If a VC is transmitting one cell for every two cell slots, it is using 50% of the link capacity, or link bandwidth.

2.1 Objective

Let S be the following set:

$$S = \{VC_i \mid VC_i's\ Quality\ of\ Service\ constraints\ are\ met\} \tag{1}$$

Our central objective is to maintain the Quality of Service desired by each VC while efficiently utilizing network resources. Let each VC_i have a peak rate denoted by $peak\ rate_i$, and each VC_i have a maximum burst size denoted by $maximum\ burst_i$. Assume all buffers have the same capacity and all links have the same bandwidth.

The following two equations state the central objective of maximizing the multiplexing gain while preserving the QoS of the involved VCs.

$$max_{\{i|VC_i \in S\}} \sum peak\ rate_i / link\ speed \tag{2}$$

$$max_{\{i|VC_i \in S\}} \sum maximum\ burst_i / buffer\ size \tag{3}$$

If Equation 2 (Equation 3) is equal to 1, then no statistical multiplexing gain has been achieved in terms of link capacity (buffer space). The greater (above 1) the value of Equation 2 (Equation 3), the larger the increase in statistical sharing of link capacity (buffer space).

Cell level QoS vs burst level QoS. Several Quality of Service (QoS) parameters which users use to indicate their desired quality of service have been defined [1]. They include the following.

1. Cell Loss Ratio. This is the ratio of cells which are initially transmitted by the source but not delivered to the destination.
2. Cell Transfer Delay. This measures the elapsed time for a cell between the network entry point and the network exit point. It includes the cell propagation delay, transmission, switching, queuing and routing delays.
3. Cell Delay Variation. This measures the jitter between consecutive cells. It is a measure of variance of the Cell Transfer Delay.
4. Peak Cell Rate. This is the inverse of the minimum interarrival period between any two consecutive cells.
5. Sustained Cell Rate. This is the averge long-term rate.
6. Burst Tolerance. This is the maximum burst size which can be sent at peak rate.

As mentioned before, the particular type of traffic we consider is continuous media traffic. How meaningful are the above QoS attributes to this type of traffic ? For instance, Cell Delay Variation measures the jitter between consecutive cells. This measure does not directly map into the jitter between bursts which, in terms of jitter, is the important metric to continuous media traffic types. Another typical QoS metric is Cell Loss Ratio. This again does not map directly into the ratio of bursts which are affected by cell losses. For instance, say VC_i has a 5% cell loss ratio. Say each burst in VC_i consists of 50 cells and that the lost cells are evenly spaced throughout VC_i's cell stream; each lost cell is followed 19 cells which are not lost. Then 100% of the bursts would exhibit loss; the burst loss ratio would be 1!

In order to be able to ensure the QoS of VCs, the QoS paramenters themselves must be meaningful measures for the particular type of traffic. In the following we re-define several QoS paramenters with respect to continuous media traffic. These are the QoS metrics we will use throughout the study.

1. Burst Peak Rate. This is the maximum burst size divided by the burst duration.
2. Burst Delay Variation. This measures the elapsed time from when the first cell from a particular burst arrives at the destination to the time when the first cell from the next consecutive burst arrives at the destination.
3. Burst Loss Rate. This is the ratio of bursts which are initially transmitted by the source but lose cells on the way to the destination.

Cell Transfer Delay, Sustained Cell Rate, and Burst Tolerance are still relevant to continuous media traffic.

In our real-time scheduling approach, we incorporate the notion of burst-level scheduling.

2.2 Traffic Control

The dual leaky bucket mechanism has been proposed as a means for traffic control (or traffic shaping) [21, 22]. It has been formulated in [1] as the Generic Cell Rate Algorithm (GCRA). The GCRA is an algorithm which defines and maintains the relationship between Peak Cell Rate, Cell Delay Variation, Sustained Cell Rate and Burst Tolerance. Conceptually, the GCRA describes two leaky buckets in tandem, i.e., a dual leaky bucket. The leaky bucket responsible for directly controlling outgoing traffic functions as a peak rate controller. This controller ensures a minimal number of cell slots between any two consectutive cells, i.e., a bound on the Peak Cell Rate and Cell Delay Variation. The leaky bucket which cells must go through before they reach the peak rate controller is a token leaky bucket bucket which regenerates tokens at a pre-specified rate, i.e., the Sustained Cell Rate, and has a bounded capacity on the number of tokens simultaneously allowed in the bucket, i.e., the Burst Tolerance. This token leaky bucket bounds the number of cells which may be transmitted at the Peak Cell Rate. It also ensures that the long-term cell rate, or Sustained Cell Rate, is the same as the pre-specified token regeneration rate. Each cell must grab a token at the token bucket, or if the token bucket is empty, wait for a new token to be regenerated, and then wait to be admitted to the network by the peak rate controller.

It is implicit in the proposed dual leaky bucket scheme that each VC will have a set of declared paramenters - Peak Cell Rate, Cell Delay Variation, Sustained Cell Rate and Burst Tolerance - based upon its predicted traffic shape. This is a straight forward way of maintaining the traffic shape throughout the network. However, the traffic shape of continuous media traffic, particular digital video, is not naturally captured by a set of static paramenters. For instance, MPEG [16], which is considered to be the most likely used digital video compression standard of the future, produces data which is highly bursty. Depending on the MPEG encoding parameters, for every fixed number of frames, or bursts, there will be a very large frame (i.e., or in MPEG terminology, an I frame) say every 16 frames. The Burst Tolerance parameter should be set to the size of this frame. The Burst Tolerance, maximum number of tokens in the token bucket, corresponds to the number of buffer slots guaranteed for that VC at each hop along its path. Given the real-time nature of continuous media traffic, i.e., very large bursts must be delivered in the same fixed time period as much smaller bursts, a user must declare worst case values for the parameters, eventhough the worst case may occur in only a small fraction

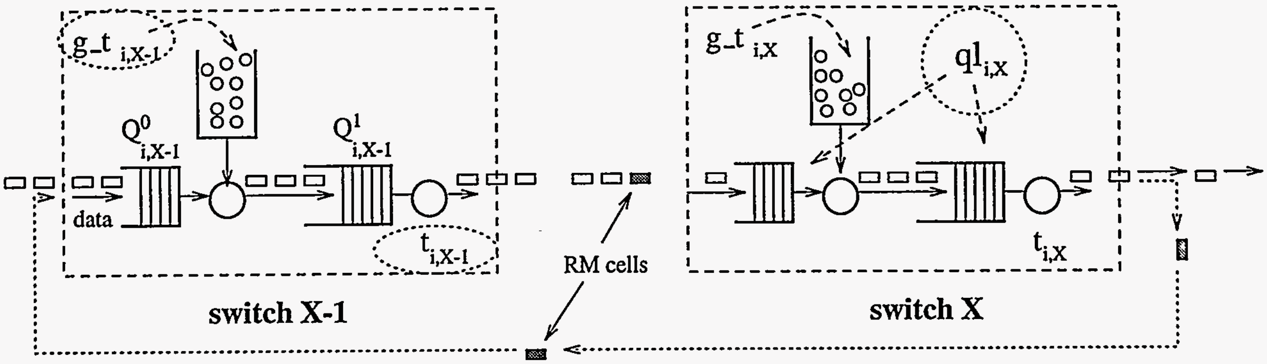

Figure 2: Adjacent switches in VC_i

(e.g., 1/16 = 6.25%) of the VC's traffic. Based upon these parameters alone, the network must decide which calls to admit/reject. The assignment of worse case parameters implies that a conservative admission control policy will admit fewer calls and hence the network will exhibit low network utilization. Or, a more liberal admission control policy will admit a greater number of calls and when faced with normal statistical fluctuations in the network traffic risk denying network resources to VCs which require some degree of guaranteed service.

To ameliorate these situations, we propose a *Shared Guaranteed Resource Dual Leaky Bucket* mechanism. In the following, we describe this mechanism in terms of the user specified and system-specified parameters.

Each VC, VC_i, corresponds to a multiple hop path, with the following user specified paramenters.

1. g_t_i corresponds to the guaranteed transmission rate of VC_i at every switch along its path. It corresponds to the Sustained Cell Rate of VC_i; the rate at which tokens arrive to fill the bucket.

2. g_buf_i corresponds to the guaranteed number of buffer slots for VC_i at every switch along its path. It corresponds to the maximum allowable number of tokens in the bucket, Burst Tolerance, for VC_i.

These guaranteed resources are particularly important to the delivery of continuous media traffic. Continuous media traffic relies on the regular delivery of a minimal amount of data in a stream-like manner.

Each VC, VC_i, also corresponds the following system controlled paramenters.

1. t_{iX} corresponds to the transmission rate of VC_i at switch X. This corresponds to the rate of the peak rate controller leaky bucket (Peak Cell Rate). When VC_i is transmitting at or above its guaranteed rate, $t_{ij} \geq g\_t_i$. When VC_i is transmitting below its guaranteed rate, t_{ij} may be set below g_t_i in order to fully share the rate (or transmission) resource.

2. ql_{iX} corresponds to the queue length (or the number of buffer slots taken by VC_i)) at switch X. There are two (logical) buffers for each VC_i: Q^0_{iX} denotes the buffer where cells are stored before they are forwarded (assigned a token), Q^1_{iX} denotes the buffer where cells are stored before they are forwarded out of the output port (see Figure 2); $|ql_{ij}| = |Q^0_{ij}| + |Q^1_{ij}|$. VC_i must be guaranteed at each switch along its path access to at least g_buf_i buffer slots. However, if VC_i is consuming less than g_buf_i slots at a node, its un-consumed slots may be used by other VCs. Again, in order to allow full sharing of the buffer resources.

Since we are dealing with the transmission of real-time traffic, it is desirable to reduce buffering (and hence delays) as much as possible. Ideally, it would be best to always set the rate such that no buffering ever occurs; no delays would ever be incurred. However, because of our objective (Equation 2) of statistical sharing of the rate resource, each VC may not be able to simultaneously 'grab' as much rate resource as it desires. Examining Switch X of Figure 2, what are the effects of setting the rates $t_{i,X-1}$ and $t_{i,X}$ on switch X's buffer occupancy ?

- Q^0_{ij}.

  1. $t_{i,j-1} \leq g\_t_i \rightarrow |Q^0_{ij}| = 0$.
  2. $t_{i,j-1} > g\_t_i \rightarrow$

  $$|Q^0_{ij}(t)| \leq t * (t_{i,j-1} - g\_t_i) \tag{4}$$

  As t increases and/or $t_{i,j-1}$ increases, $|Q^0_{ij}(t)|$ increases.

- Q^1_{ij}.

  1. $g\_t_i > t_{ij}$ is not feasible. g_t_i is the pre-agreed upon rate which the network guaranteed to VC_i, so the minimal value of t_{ij} (for all j) must always be at least g_t_i.
  2. $g\_t_i \leq t_{ij} \rightarrow$

  $$|Q^1_{ij}| \leq g\_buf_i * (1 - t_{i,j}) < T\_buf_X \tag{5}$$

  Since, our scheme supports full buffer sharing, $|Q^1_{ij}|$ is bounded above by the total number of buffer slots available at output port X, T_buf_X.

Thus we can see that the appropiate manipulation of rates are important not only for meeting real-time constraints but also for controlling buffer occupancy. This is true for Equation 4. Equation 5 is bounded above by a pre-defined constant.

Admission Control. We assume the following admission control policy is enforced. The network and the user negotiate the following:

- Deterministic guarantees for pre-agreed upon resources. The network agrees to provide guaranteed service specified by the parameters (g_buf_i, g_t_i), at every switch along VC_i's path.

- Statistical guarantees. The network will also only admit VC_i if it considers the existing network traffic and concludes that there is a high probability that it will be able to allocate resources, above the guaranteed service, at every switch along the VC_i's path. To compute the amount of resources that are above VC_i's guaranteed resources which the network must statistically guarantee, the network must examine VC_i's QoS parameters - burst peak rate, burst delay and delay variation, and burst loss ratio.

3 Rate and Buffer Control Algorithms

In this section, we present several algorithms which dynamically allocate/de-allocate the rate and buffer space resources of network switches. Rate and buffer scheduling is performed at each switch node in relation to the proposed *Shared Guaranteed Resource Dual Leaky Bucket* mechanism. The primary objective of scheduling is to maintain the real-time nature of the traffic as much as possible. In the event of resource contention, the scheduling algorithm decides the appropiate allocation of resources. In the event of heavy traffic conditions, a congestion control algorithm is invoked to detect potential congestion and react in ways to avert congestion (buffer overflow).

Desirable properties of our approach include:

- Isolation. Each VCs guaranteed resources, (g_t_i, g_buf_i), are guaranteed to be accessable to each VC regardless of fluctuating network conditions.

- Efficiency. The rate and buffer resources are fully shared. That is, if a VC is not using its guaranteed resources, another VC may use them.

- Simplicity. The algorithms are computationally simple. The proposed *Shared Guaranteed Resource Dual Leaky Bucket* mechanism requires little additional hardware functionality than the standard proposed dual leaky bucket described by the GCRA algorithm [1]. Recall that all that is needed to implement a leaky bucket (peak rate controller) is a timer and counter; a token leaky bucket requires an additional counter (in addition to its own timer).

As mentioned in Section 1, we will use the notion of the proposed Resource Management cell. An ATM cell will indicate whether it is a RM cell by setting its payload type field to 110 [1]. The following three types of RM cells will be used by this study's algorithms.

- A *burst reservation* RM cell. Most RM cells in the network will be of this type. It is assummed that a *burst reservation* RM cell will immediately precede each burst. These cells are used by the *real-time rate scheduler*. Hereafter for brevity, if the term RM cell is used, it will imply a *burst reservation* RM cell.

```
Every cycle time slots --
Repeat:
    receive RM cell;
    compute the new desired rate;
    for i = 1 to period/cycle
        $t_{iX}$ = request (desired rate);
        based upon $t_{iX}$ forward one of the following:
            (same RM cell, new RM cell, no RM cell);
        forward DATA at rate $t_{iX}$ for cycle time slots;
        compute the number of late cells;
        if next cell is an RM cell then
                break out of the for() loop;
        compute the new desired rate;
```

Figure 3: Per-VC Rate Scheduler

- A *backward no-congestion* RM cell, and a *backward congestion* RM cell. These two types of RM cells are used by the *link-link congestion control algorithm* (Section 3.3). They serve as congestion indicators. They travel at most 1 hop (upstream) and are discarded at the receiving node. Any node (besides a source node) may generate one of these types of cells.

3.1 Real-time rate scheduler

- *Per-VC.* Each VC consists of a series of Resource Management (RM) cells interspersed among the data cells. Each RM cell announces the beginning of a burst. A burst is denoted by two fields in the RM cell, (*num*,*period*), where *num* denotes the number of cells in the burst, and *period*, denotes the burst duration (number of cell slots).

- *Essential idea of algorithm.* As a burst appears at the edge of the network, it will propagate through the network. In order to prevent too many buffer slots from being occupied by cells from this burst (and delays from accruing), the network increases the rates, t_{iX}'s, along the path which the burst is expected to propagate through. The desired increase in rate is computed by examining the (*num*,*period*) fields in the immediate RM cell as well computing the number of delayed cells from the previous burst.

- *Description of algorithm.* Each VC will have a *Per-VC Rate Scheduler.* This scheduler will execute every fixed number of time slots, *cycle* time slots. Each *Per-VC Rate Scheduler*

```
Every cycle time slots --
    1. if the summation of all the requested rate < 1
           return $t_{iX}$ to $VC_i$ ;
    2. else
           sort (prioritize) the VCs in decreasing order of requested
               desired rates (relative to each VC's guaranteed rate);
           distribute in a greedy manner the rate resource to the
               sorted VCs;
```

Figure 4: Rate Resolution Scheduler

will receive RM cells, retrieve information from the RM cells, and attempt to satisfy the transmission of the 'requested' burst by adjusting its rate request to equal *num/period.* If there are 'late' cells from other cycles, it will request a high rate. The *Per-VC Rate Scheduler* will send the computed rate request to the *Rate Resolution Scheduler*, adjust its rate to the rate assigned by the *Rate Resolution Scheduler* and then forward at the beginning of the cycle either no RM cell, the same RM cell, or a new RM cell. Figure 3 depicts a the skeleton code for the *Per-VC Rate Scheduler.* The Appendix contains the detailed code.

If the RM cell requests a rate which is greater than the guaranteed rate, it may or may not receive it. If no other VC is using (or requesting) the additional rate, then the VC will be allocated the additional rate. If another VC is using the additional rate (in excess of its own guaranteed rate) then a contention resolution algorithm will be used. The rate resource will be allocated to the VC with the 'earliest' deadline (in terms of largest relative number of delayed cells per burst, i.e., relative requested desired rate) in a 'greedy' manner. Figure 4 depicts a the skeleton code for the *Rate Resolution Scheduler.* The Appendix contains the detailed code.

Whenever a rate is assigned (by the rate scheduler), cells will be transmitted via time constrained rate control.

Definition 1 Time constrained rate control *occurs when a VC is given a fixed number of cells to transmit, denoted by num, and a fixed number of cell slots in which to transmit, denoted by cycle, and it transmits one cell every cycle/num time slots.*

An example of rate contention is shown in Figure 5. In this Figure, two streams, A and B, are simultaneously contending for the rate resource; i.e., streams A and B will be

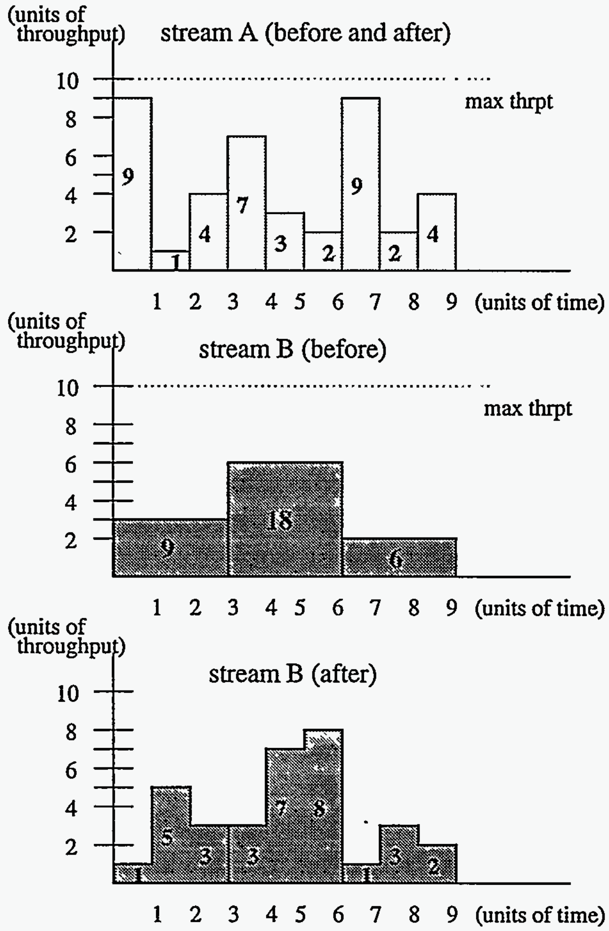

Figure 5: Example of EXPLICIT ALGORITHM 1

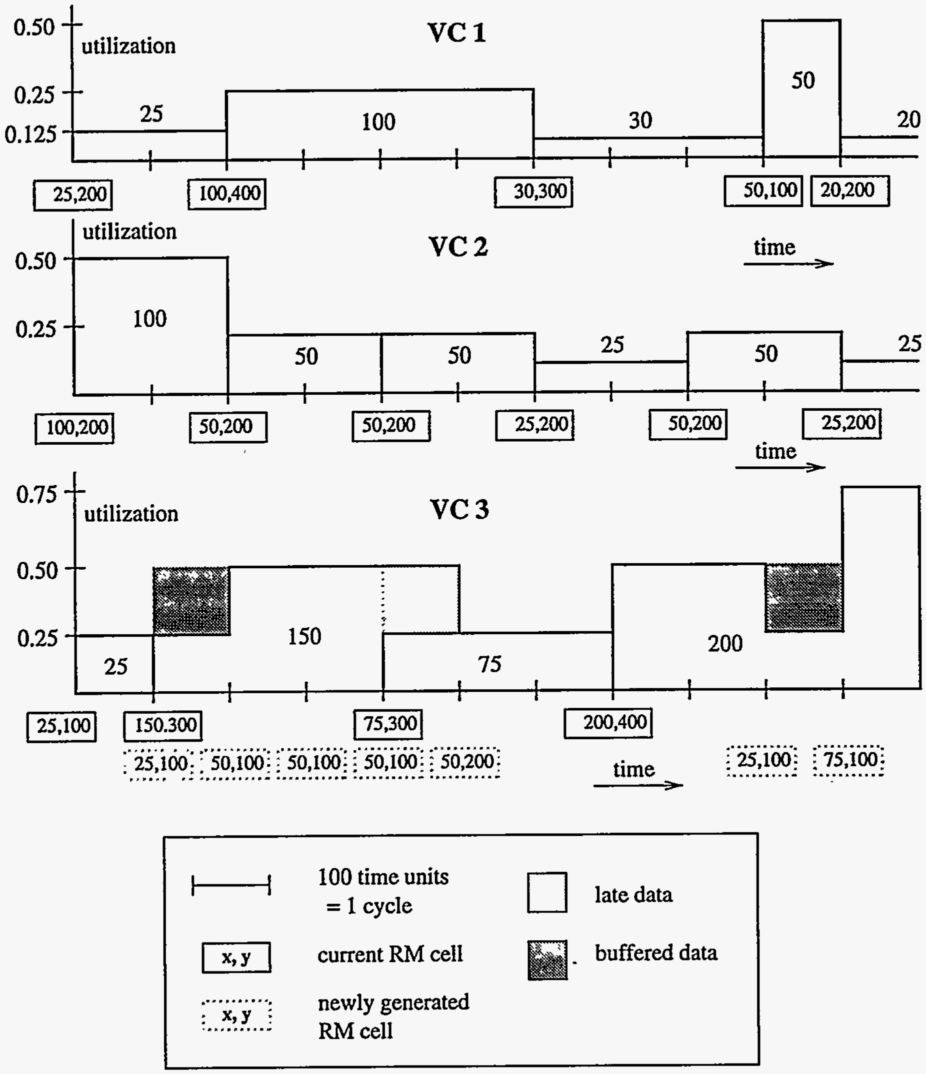

Figure 6: Contention resolution

multiplexed together. At time unit 0, a RM cell from stream A arrives with $(9, 1)$, which denotes that it is requesting to send 9 units of data in the immediate next time unit. Also at time unit 0, a RM cell from stream B arrives with $(9, 3)$, which denotes that it is requesting to send 9 units of data in the immediate next 3 time units. Only 10 units of data may be scheduled during any single time unit. Since stream A obviously has the 'earliest' deadline, it is allowed to transmit 9 units of data in the next time unit. Stream B thus must buffer 2 units of data. Now at time unit 1, stream B would like to transmit 5 units of data in the next time (2 buffered from the previous time unit and 3 which will be arriving immediately). This is considered 'greedy' because, stream B would also be able to meet its time deadline if it transmitted 4 units of data each in the next 2 time units. In time unit 1, it is able to transmit all 5 units of data because stream A is only requesting 1 unit of data per unit time. Examining stream B, after having been multiplexed with stream A, we note that, even though its traffic shape has changed slightly, all of its bursts still arrive within their deadlines.

Figure 6 depicts 3 contending VCs - VC_1, VC_2 and VC_3. Contention occurs at time cycles $1, 2, 3$, and 9. As can be seen by VC_3, as the shape (rate) of the traffic changes, a new RM cell must be generated in order to inform downstream switches of the change. If there exists contention, the contention resolution algorithm determines the rate, via a priority scheme, that contending VCs will be assigned. All three VCs have a guaranteed rate of 0.25 each.

Synchronization. An assumption that the algorithm makes is that time is discretely divided into *cycles*, where each *cycle* consists of a fixed number of time slots. RM cells must always be inter-spaced among data cells (per-VC) in an integral multiple of *cycle* slots (see Figure 6). Thus it is assumed that initially when a VC enters a network it must be synchonized with other VCs by being buffered a maximum of *cycle* time slots. **Once the VC has been synchronized (delayed) with other contending VC's (which have already been synchronized at an earlier time) it will not require any more synchonization delays (buffering) at downstream switches.** Additional synchronization delays will only occur if a VC, say VC_i, goes through a switch(es) which only has VCs which use completely disjoint paths from all upstream switches in VC_i's path. Thus this algorithm does not imply the necessity of an inordinate amount of buffering for synchronization purposes at any single switch. The main purpose for the synchonization is that the algorithm (with or without contention resolution) must be invoked at the beginning of each cycle.

3.2 Per-Output Port Buffer Scheduling Algorithm

This algorithm supports full buffer sharing with isolation (guaranteed Burst Tolerance).

- *Description of Algorithm.* Each VC_i is guaranteed access to at least g_buf_i buffer slots at each switch along its path. If a VC needs more buffers slots (due to an increased rate from its upstream node), it can 'take' as much as it needs. However, it can not take resources from another VC, say VC_j, if VC_j is using all of its g_buf_j buffer slots. If VC_j requires less than g_buf_j buffer slots, then its unused buffer slots may be taken by another VC (to support full sharing).

When the entire buffer space is filled, and new cells arrive from an upstream node, a VC is chosen on a round-robin basis as the one to lose cells. All possible cells from this VC which belong to the same burst are discarded. This is preferable to discarding the same number of cells from different bursts since we are attempting to preserve QoS at the burst level. The Appendix contains the detailed code for this algorithm.

3.3 Link-Link Congestion Control

As long as peak resources are not allocated, there is always the possibility of congestion. We define a potential congestion point to refer to an output port where the outgoing rate is close to 1 cell per cell slot and the buffer occupancy is past a pre-specified threshold, say 85%.

- *Description of the Algorithm.* The congestion control algorithm continuously executes at each output port. It checks for potential congestion. If potential congestion is detected, a *backward congestion* RM cell is sent to the appropiate *VC*s neighboring upstream nodes (see Figure 7). The receipt of a *backward congestion* RM cell tells the receiving node to decrease the rate of its VC(s) (to their guaranteed rate(s)) which are transmitting to the downstream congested node. Once congestion has been detected, backpressure will cause all the nodes from all contributing VCs to decrease their rates. Similarly, once the congested node's queue length has decreased past a pre-specified threshold, the congestion control algorithm will detect the passing of a congestion state and generate a *backward no-congestion* RM cell which will signify to the upstream nodes that they can once again increase their rates. Again backpressure will cause the generation of *backward no-congestion* cells to propagate to all more upstream nodes. The Appendix contains the detailed code for this algorithm.

3.4 Discussion of Algorithms

All of the algorithms proposed in the previous section execute in order constant time except for the *Rate Resolution Scheduler* which is part of the *Real-time Rate Scheduler* (Section 3.1). The *Rate Resolution Scheduler* must sort the contending VCs in order to distribute the rates among them. Its complexity is order $NlogN$ where N is the number of contending VCs. The implicit assumption is that the algorithms must execute in less than 1 time slot. For an OC-3 155 Mb/s link, a time slot is approximately 2.75 microseconds. It is expected that the algorithms will be implemented in firmware so the complexity of the algorithm only becomes an issue for a very large number of contending VCs on a very high speed network. If the execution time of the algorithm is an issue, modifications may be made such as the following. (a) Increase *cycle* so the algorithm is invoked a fewer number of times. Each time it is invoked it will incur a delay (assumming it cannot execute in less than 1 time slot). That delay should be approximated and computed along with the number of hops in the path. The effect on QoS should be computed when deciding whether to accept/reject a VC. (b) The maximum number of possible VCs, such that the algorithm can be executed in less than 1 time slot, should be computed. Any additional VCs which are admitted to the network must be 'bundled' with an existing VC. That is the more than 1 bundled VC should logically behave as 1 VC; each RM cell will announce a burst which would be a burst consisting of cells from more than 1 VC.

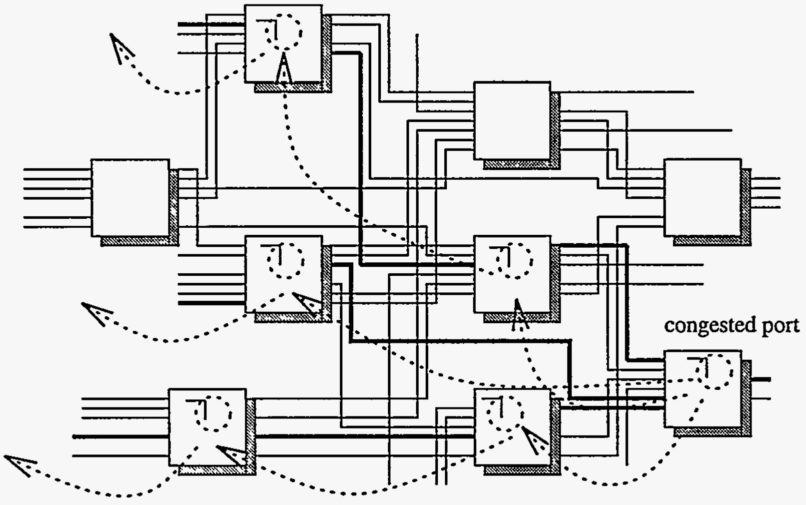

Figure 7: Congested output port and affected VCs

4 Buffering Strategy Analysis

In this section, we discuss buffering at the cell level and buffering at the burst level. In terms of buffering at the cell level, we discuss the effect of cell spacing. A form of cell spacing which we defined in Section 3, and which the *Real-time Rate Scheduler* implements, is *time-constrained rate control.* The objective of cell spacing is to decrease unnecessary queueing by not transmitting cells in consecutive time slots. Decreasing queueing implies an increase in statistical multiplexing gain while potentially still meeting the QoS requirements of all additional and existing VCs.

Example 1 *Consider two bursts which arrive simultaneously. Each burst duration is 8 cell slots. Let Burst* $1 = \{A, B, C, D, \emptyset, \emptyset, \emptyset, \emptyset\}$*, Burst* $2 = \{E, F, G, H, \emptyset, \emptyset, \emptyset, \emptyset\}$*, Burst* $3 = \{A, \emptyset, B, \emptyset, C, \emptyset, D, \emptyset\}$ *and Burst* $4 = \{\emptyset, E, \emptyset, F, \emptyset, G, \emptyset, H\}$*. When Burst 1 and Burst 2 are multiplexed on the same output port, assuming a round robin scheduler, and Burst 3 and Burst 4 are multiplexed on another output port:*

	Burst 1 and Burst 2		Burst 3 and Burst 4	
	outgoing cell	queued cells	outgoing cell	queued cells
output slot 1 :	A	E	A	none
output slot 2 :	E	B, F	E	none
output slot 3 :	B	F, C, G	B	none
output slot 4 :	F	C, G, D, H	F	none
output slot 5 :	C	G, D, H	C	none

```
output slot 6 :   G            D, H         G            none
output slot 7 :   D            H            D            none
output slot 8 :   H            none         H            none

Maximum queue length = 4;                   Maximum queue length = 0;
Average queue length = 2.                   Average queue length = 0.
```

Given N bursts denoted by $burst\ 1, burst\ 2, ..., burst\ N$. Let $burst\ i$ consist of N cells: $c_{i1}c_{i2}...c_{iN}$.

Definition 2 *N bursts arrive in* **burst form** *if on the first output slot, cells $c_{11}, c_{21}, ..., c_{N1}$ arrive simultaneously, and each cell from each burst arrives in the cell slot immediately following its preceding consecutive cell.*

Definition 3 *A burst, burst i, is in* **spaced form** *if there exists an empty slot between $c_{i,j}$ and $c_{i,j+1}$ for some j.*

Note that time-constrained rate control produces bursts in *spaced form.*

Since our study focuses on real-time traffic, we only consider spacing techniques which either output cells at the same rate as a non-spaced technique, or which would output cells in a manner that would not violate their real-time constraints (e.g., time-constrained rate control). Obviously, decreasing queueing is trivial if one is allowed unlimited buffering delays.

Theorem 1 *Let B denote a set of N bursts where each burst is in burst form. Let SB denote a set of the same N bursts where each burst is in spaced form. When the bursts from set B and set SB are each multiplexed onto an outgoing link, the average and maximum queue length associated with set B will always be strictly greater than the average and maximum queue length associated with set SB. Both sets output cells at the same rate.*

Proof. For each set B and SB, sort the N bursts, $burst\ 1, burst\ 2, ..., burst\ N$ such that $|burst\ i| > |burst\ (i+1)|$. The superscripts B and SB will be used to distinguish between the two sets of bursts. In cases where a superscript is not used, the case is applicable to either set.

Let (a) $sum = \sum_{i=1}^{N} |burst\ i|$ (b) $q[i]$ denotes the number of queued cells at the ith output slot. For instance, $q^B[1] = N - 1$ and $q^B[sum] = 0$. (c) $count[i]$ denotes the summation of the number of times cell i contributed to the queue length during each contention time slot. Note that: $\sum_{\forall x,y} count[c_{xy}] = \sum_{j=1}^{sum} q_j$. Also, the average queue length is $(\sum_{i=1}^{sum} q_i)/N$, and the maximum queue length is $max_i q_i$.

It follows that:

$$count^B[c_{xy}] \geq count^{SB}[c_{xy}], \quad \forall\ x, y \tag{6}$$

Since bursts in the set SB must be spaced, there exists at least one c_{xy} such that: $count^B[c_{xy}] > count^{SB}[c_{xy}]$. Thus the average queue length of set SB is strictly less than the average queue length of set B.

Now we must show that the maximum queue length associated with set B is always *strictly* greater than the maximum queue length associated with set SB.

Let the pivot index, *pivot*, occur at index i such that $q[i] > q[i+1]$. The pivot occurs at the time slot before the time slot which has no incoming cells. Set B has only one pivot index where $pivot^B = |burst\ 1|$ and

$$q^B[pivot^B] = \sum_{i=2}^{N} |burst\ i| \tag{7}$$

Also note that:

$$q^B[i] = q^B[i-1] + 1, \quad \forall\, i \leq pivot^B \tag{8}$$

Since all bursts in set SB are spaced, $pivot^{SB} > pivot^B$. Using Equations 7 and 8,

$$q^{SB}[pivot_i^{SB}] < q^B[pivot^B], \quad \forall\ pivot^{SB} \tag{9}$$

Note that set SB may have more than one pivot index. Thus we have shown that maximum queue length of set SB is strictly less than the maximum queue length of set B □.

Buffering at the burst level immediately entails the necessity for appropiate buffer dimensioning. Usually the amount of buffer resources are known, and the amount of buffering required to be reserved for a VC must be computed based upon its expected burstiness.

We propose a buffer dimensioning procedure which assumes the following.

- mbs_i corresponds to the maximum burst size of VC_i. The user may specify the peak burst size or the average maximum burst size. For example, say VC_i consists of the following pattern of bursts: 30, 10, 11, 10, 35, 12, 12, 10, 25, 11, 10, 10, where the units are the number of cells per 100 cell slots. Say also that $g\_t_{iX} = 0.1$. The user may specify $mbs_i = 25$ (peak burst size), or $mbs_i = 20$ (average burst size), depending on the QoS expected. For a high quality of service, a user would usually specify a value near the peak burst size for the *mbs*.

- Continuous media traffic not only exhibits periodicity in the time domain alone, in terms of the appearance of variable size bursts at fixed time intervals, but periodicity may also occur in terms of a fixed range of burst sizes at fixed time intervals. For instance in MPEG, I frames, frames (bursts) which are much larger than other frames, occur at fixed time intervals. A typical I frame ratio would be 1 I frame every 12 or 16 frames.

  Ib_i corresponds to the number of intervals between the occurences of maximum bursts. For instance, in the above example, $Ib_i = 3$. If there are no set of bursts which are distinctly larger and occur at fixed periodic time intervals, $Ib_i = 1$.

The procedure includes the following steps.

1. Compute mbs_i and Ib_i for all contending VC_i's as a function of the expected QoS per VC.

2. Compute the collision probability.

   Let X be a random variable which denotes the number of bursts which may occur simultaneously. Then $P(X=0) = \prod_{i=0}^{N-1}(1-1/Ib_i)$, $P(X=1) = \sum_{i=0}^{N-1} 1/Ib_i \prod_{j=0,j\neq i}^{N-1}(1-1/Ib_j)$, $P(X=N-1) = \prod_{i=0}^{N-1} 1/Ib_i$, and in general:

$$P(X=q) = \sum_{i=0}^{N-1} \sum_{j=i+1}^{N-1} \dots \sum_{z=i+q-1}^{N-1} (1/Ib_i)(1/Ib_j)\dots(1/Ib_z) \prod_{a\neq i,j,\dots,z;a=0}^{N-1} (1-1/Ib_a) \quad (10)$$

3. Compute the weighted maximum burst size, *w_max_burst*.

   The weighted burst size is a function of all the user specified maximum burst sizes (mbss) and the number of intervals between the occurences of maximum bursts (Ibs).

$$w\_max\_burst = \sum_{i=0}^{N-1} (mbs_i/Ib_i) \quad (11)$$

4. Compute the number of buffer slots as a function of desired burst loss ratio (of the VCs sharing the buffer space).

   - Find the largest CL such that:

$$burst\ \ loss\ \ ratio \leq \sum_{i=CL}^{N} P(X=i) \quad (12)$$

   The *probabilistic average* number of buffer slots necessary is $(CL-1)*w\_max\_burst$. If a more stringent QoS guarantee is necessary, then the *probabilistic worst case* number of buffer slots necessary is $\sum_{i=0}^{CL-1} mbs_i$, where mbs_i's are sorted in non increasing order, i.e., $mbs_i \geq mbs_{i+1}$.

5 Simulation

The simulation was used to demonstrate the efficacy of the proposed algorithms described in the previous section.

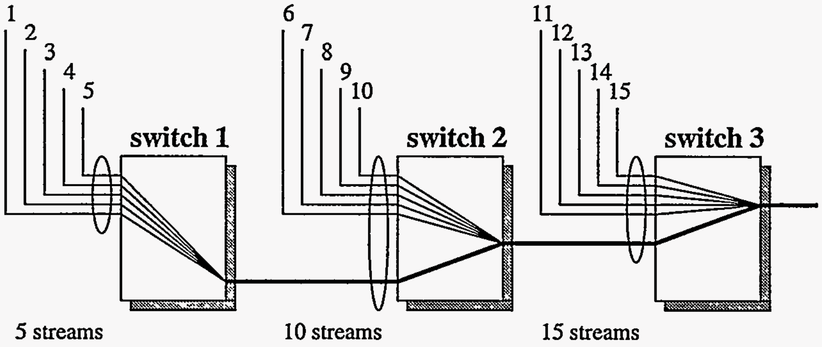

Figure 8: The parking lot configuration

5.1 Description of Simulation Model

The Ptolemy simulation tool [17, 18, 19], developed at UC Berkeley, was used to implement our models. Ptolemy provides support for a wide variety of computational models, called *domains*, such as dataflow, discrete-event processing, communicating sequential processes, computational models based upon shared data structures and finite state machines.

Our model was developed in the Discrete Event (DE) domain. In Ptolemy, the DE domain provides a generic discrete event modeling environment for time-oriented simulations of systems such as queueing models and communications models.

5.1.1 Block Diagram for Internal Switch

Figure 9 depicts the block diagram of a switch output port used in the simulation model. The Figure shows the internal switch mechanisms which provide resource allocation/de-allocation to VCs. These mechanisms are at each output port of a switch. Recall that the only significant variable delay involved in network transmission is the queueing delay, and we previously assumed an output buffered switch architecture.

Initially as cells arrive at the output port, they are demultiplixed via their VC identifier in the ATM cell header. Each VC has a logical queue which uses the FIFO scheduling discipline. Each VC has an associated guaranteed amount of buffer space. The amount of buffer space is negotiated at call setup. Cells are only buffered if their 'Per-VC Rate Scheduler' (Section 3.1) is not idle, and either their associated guaranteed amount of buffer space is not full or other VCs are not currently using their guaranteed buffer space, i.e., full buffer sharing is supported. These actions, buffer allocation/de-allocation is performed by the 'Per-Output Port Buffer Scheduler' (Section 3.2). Each VC has a 'Per-VC Rate Scheduler' which is responsible for serving (transmitting) cells at a certain rate. This rate is first computed using the information from each newly received RM cell as well as information about previously delayed cells. The

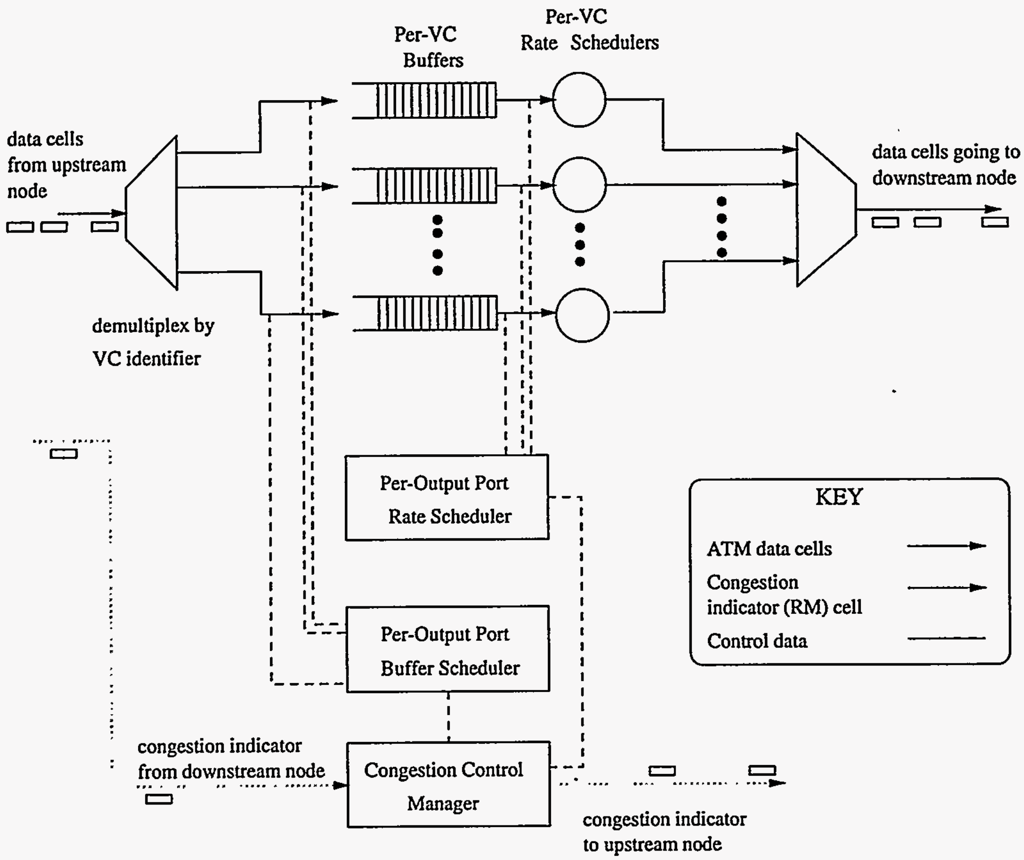

Figure 9: Block diagram of switch internals

24 Mb/s
buffer = 10000
period = 41.67 msec
	FIFO	FIFO with rate control	EXPLICIT

+/- .4167 msec	60.43%	62.30%	76.74%
+/- 4.167 msec	91.44%	92.25%	93.32%

(a)

24 Mb/s buffer = 900	FIFO	FIFO with rate control	EXPLICIT
# lost bursts switch 2	290 (8.06%)	44 (1.28%)	37 (1.03%)
# lost bursts switch 3	98 (1.81%)	61 (1.13%)	14 (0.26%)

(b)

Figure 10: (a) burst interarrival delays, (b) burst losses

'Per-VC Rate Scheduler' then 'requests' the ideal computed rate from the 'Per-Output Port Rate Scheduler' (Section 3.1). This scheduler is responsible for resolving contention when the sum of the rates requested is larger than the capacity of the outgoing link. The congestion control manager implements the congestion control routines in Section 3.3. It is mainly responsible for setting and clearing congestion flags.

5.1.2 Overall Model

For the overall model, the parking lot model was used (see Figure 8). It is an especially useful model because it can be used to observe the effect.of increased contention at each hop.

In our model, at each stage (switch), five additional sources are multiplexed onto a single outgoing link. The first switch multiplexes 5 sources; the second switch multiplexes another 5 sources with the output from the first switch; the third switch multiplexes an additional 5 sources with the output from the second switch (Figure 8).

Input traces. The input streams consisted of (simulated) frames from a MPEG codec. The input video stream for the MPEG codec was a 3 minute 40 second sequence from the movie Star Wars [16]. The sequence was digitized from laser disc with a frame resolution (similar to NTSC broadcast quality) of 512 × 480 pixels. This particular Star Wars sequence was chosen because it contained a mix of high and low action scenes. The interframe to intraframe ratio was 16. The quantizer scale was 8. For these parameters, the image quality was judged to be good (constant) through the entire sequence of frames. The coded video was captured at 24 frames/second. Every period, frame or burst interarrival period, was 41.67 milliseconds.

24 Mb/s infinite buffer	FIFO	FIFO with rate control	EXPLICIT
switch 1			
average	69	290	290
maximum	758	771	910
minimum	0	0	98
variance	18.7E3	30.7E3	30.7E3
switch 2			
average	333.5	373	373
maximum	1111	834	968
minimum	0	0	155
variance	113E3	46.4E3	46.4E3
switch 3			
average	437	296	301
maximum	1158	947	1110
minimum	0	0	98
variance	65E3	31.7E3	32.9E3

Figure 11: Queue length statistics

We examine the following cases.

1. **FIFO.** In this case, all switches provide first-in-first-out scheduling. There is no notion of guaranteed rates and/or guaranteed buffer slots per VC. The rate and buffer resources are dynamically allocated/de-allocated according to the FIFO discipline.

2. **FIFO with time-constrained rate control.** In this case, the rate resource is computed for each burst according to the values found in the immediately preceding RM cell. There is no notion of guaranteed rates and/or guaranteed buffer slots per VC. For brevity, hereafter we will refer to this case as **FIFO with rate control.**

3. **EXPLICIT scheduling.** This case uses all the algorithms found in Section 3 except for the Congestion Control algorithm. The additional features this case includes over the **FIFO with rate control** is the notion of guaranteed rates and/or guaranteed buffer slots per VC, and the scheduling of buffer and link capacity according to burst-level QoS.

4. **EXPLICIT scheduling with congestion control (CC).** This case is the same as the above **EXPLICIT scheduling** case with the link-link congestion control algorithm also implemented.

5.2 Results

In this section, we present through simulation results which show how well the above four schemes are able to maintain the QoS of the VCs. The Ptolemy simulation tool developed at UC Berkeley was used to implement the model. Each test was run for 15 seconds.

20 Mb/s infinite buffer	EXPLICIT w/out congestion control	EXPLICIT with congestion control
switch 1		
average	292.5	292.5
maximum	836	836
minimum	0	0
variance	30.4E3	30.4E3
switch 2		
average	367.4	4670
maximum	941	19060
minimum	0	0
variance	44.8E3	33E6
switch 3		
average	16700	14680
maximum	25000	21710
minimum	0	0
variance	63E6	38.5E6
delays		
+/- 2.08 msec	74.33%	73.26%
+/- 6.25 msec	89.3%	87.96%
+/- 10.42 msec	95.18%	94.65%

Figure 12: Congestion control statistics

- **Burst delays.** A burst interarrival delay is the elapsed time from when the first cell is output by switch 3 to the time when the first cell from the next consecutive burst is output by switch 3. Figure 10 (a) depicts the burst interarrival delay statistics. Recall that every period is 41.67 milliseconds. 76.74% of bursts arrive within $+/-0.4167$ milliseconds of the deadline using the EXPLICIT scheme. Also using the EXPLICIT scheme, 93.32% of bursts arrive within $+/-4.167$ milliseconds of the deadline. In all schemes, virtually all the bursts arrive within $+/-8.334$ milliseconds.

  The performance of the FIFO with rate control is sightly better than the performance of the FIFO scheme. The EXPLICIT scheme performs much better than either of the other schemes because its scheduling algorithm works on a per-burst basis, where bursts which have the greatest number of delayed cells are given the higher priority.

- **Burst losses.** A burst is considered lost if any cell in the burst is loss. In this test, the performance of the EXPLICIT scheme is slightly better than the performance of the FIFO with rate control scheme. The FIFO scheme performs much poorer than either of the other schemes. This can be attributed to its much larger queue lengths at switch 2 and switch 3 which can be attributed to the burstier (non rate-controlled) traffic (see below part on queue lengths). It is of interest to note that in terms of cell loss, the EXPLICIT scheme had a much higher cell loss ratio than the other two schemes. This can be attributed to the way in which cells are discarded by the *Per-Output Port Buffer Scheduling Algorithm*; in the event of buffer overflow, when a VC is chosen (round-robin manner) as the VC to lose cells, all cells from that VCs same burst are discarded, i.e., multiple cells per buffer overflow event are discarded. In the other two schemes, a single cell is discarded for every buffer overflow event. Burst loss is near 0 at switch 1 for all three schemes since there is little contention. Figure 10 (b) depicts the burst losses for the three schemes.

- **Queue length.**

  Figure 11 summarizes the queue length statistics for all three switches.

  - **Switch 1.** Both the FIFO with rate control scheme and the EXPLICIT scheme show much greater queueing than the FIFO scheme. The sources initially send bursts consisting of back-back cells; there is no smoothing at the source. Thus when the cells arrive at the switch, in the FIFO with rate control scheme and the EXPLICIT scheme, cells must be buffered because each VC's rate is controlled. However in the FIFO scheme, cells are sent out of switch 1 as soon as possible.
  - **Switch 2.** The maximum queue length and the variance of the queue length is significantly greater for the FIFO scheme because the outgoing traffic from switch 1 is much burstier than the outgoing (rate controlled) traffic from the other two schemes.
  - **Switch 3.** The average queue length and the variance of the queue length is significantly greater for the FIFO scheme, again, because the outgoing traffic from switch 2 is much burstier than the outgoing (rate controlled) traffic from the other two schemes.

- **Congestion control.** Figure 12 depicts the queue length statistics and delays for the EXPLICIT scheme and the EXPLICIT scheme with link-link congestion control. Both schemes use the same parking lot model as the other tests but in this test all the outgoing links have a capacity of 20 Mb/s (instead of 24 Mb/s as in the previous tests). We restrict the bandwidth in order to ensure a congested state. Buffer sizes at all switches are infinite so there are no losses. The propagation delay between hops was set to 1 millisecond.

  From Figure 12, the maximum queueing occurs at switch 3 in the case with no congestion control. In the case with congestion control, although the queueing at switch 3 is less than the maximum, the queueing at switch 2 is greater than at switch 2 in the case with no congestion control. Thus in the case with link-link congestion control, the load, or data, is load balanced among switches 2 and 3. Examining Figure 12, the delays between the two schemes are very slight; the EXPLICIT scheme without congestion control has only slightly lower delays then the EXPLICIT scheme with congestion control.

6 Results and Conclusion

This study addressed the issue of traffic control for continuous media traffic. The major points include:

- *Shared Guaranteed Resource Dual Leaky Bucket mechanism.* We proposed an extension of the currently proposed dual leaky bucket mechanism which is applicable to the support of continuous media traffic. This mechanism allows for the full sharing of rate and buffer resources, as well as real-time traffic delivery, through appropiatly adjusting the leaky bucket parameters.

- *Simple Implementation.* The proposed traffic control mechanisms (algorithms) may be implemented with relatively low overheads.

  - *Synchronization delays.* The *Real-time Rate Scheduler* algorithm (section 3.1) assumes that bursts are aligned at each switch. For this to occur, a VC will only experience a synchronization delay at the initial switch where it may be contending with other VCs; bursts will **not** have to be re-aligned at every switch. Typically, a VC will only experience a synchronization delay (a maximum of *cycle* slots) at the first switch at the edge of the network. See section 3.1 and Appendix A for details

  - *Buffer sizes.* The simulations in section 5 show that buffer size requirements are reduced noticeably when the proposed forms of rate/congestion control are implemented. The analysis in section 4 showed that buffers may be dimensioned as a function of the desired burst loss ratio and their respective burst frequency.

  - *Processing overhead.* The proposed algorithms all execute in order constant time except for the *Rate Resolution Scheduler* (part of the *Real-time Rate Scheduler*) (section 3.1). Its complexity is $NlogN$ where N is the number of contending VCs. Section 3.4 discusses how this processing overhead may be minimal, or how, as the number of VCs increases, the overhead may be minimized.

- *Rate control for real-time continuous media traffic.* As mentioned before, rate control for Real-time VBR traffic is difficult to implement due to the real-time nature of the traffic. We proposed a method of setting the leaky bucket peak rate enforcer to a rate which attempts the timely delivery of all cells in the current burst as well as gives priority to bursts which have accumulated late cells. We showed analytically and through simulation that a form of rate-control results in less queueing (buffering), and hence a larger potential statistical multiplexing gain.

- *Enforcing burst level QoS over cell level QoS.* Our proposed algorithms were designed to optimize burst level QoS, i.e., burst loss and burst delay. Scheduling is done using a type of a pseudo 'earliest deadline first' approach where bursts which have the relative greatest number of delayed cells are given the highest priority. When buffer overflow occurs, cells from bursts which have already lost cells are discarded over cells from other bursts. These scheduling and buffer management techniques result in significantly lower burst delays and losses.

References

[1] ATM FORUM, "ATM User-Network Interface Specification", Version 3.1.

[2] Bonomi, Flavio, Fendick, K., "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service", *iEEE Network*, March/April 1995.

[3] Boudec, J., "The Asynchronous Transfer Mode: A Tutorial", *Computer Networks and ISDN Systems*, Vol. 24, pp. 279-309, 1992.

[4] Boyer, P., Tranchier, D., "A Reservation Principle with Applications to the ATM Traffic Control", *Computer Networks and ISDN Systems*, Vol. 24, 1992, pp. 321-334.

[5] Ferrari, D., Verma, D., "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", *IEEE Journal on Selected Areas in Communications*, Vol 8, No 3, April 1990.

[6] Golestani, S.J., "Congestion-free Communication in High Speed Packet Networks", *IEEE Transactions on Communications*, December 1991.

[7] Gong, Y., Akyildiz, I., "Dynamic Traffic Control Using Feedback and Traffic Prediction in ATM Networks", *IEEE Proceedings of INFOCOM*, 1994.

[8] Jain, R., "Congestion Control and Traffic Management in ATM Networks: Recent Advances and A Survey", to appear in *Computer Networks and ISDN Systems.*

[9] Kataria, D., "Comments on Rate-Based Proposal", AF-TM 94-0384, May 1994.

[10] Kawarasaki, M., and Jabbari, B., "B-ISDN Architecture and Protocol", *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 9, pp. 1405-1415, Dec. 1991.

[11] Kung, H.T., Blackwell, T., Chapman, A., "Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptice Credit Allocation, and Statistical Multiplexing", *Proceedings ACM SIGCOMM*, 1994.

[12] Kung, H.T., Chapman, A., "The FCVC (Flow-Controlled Virtual Channels) Proposal for ATM Networks", *anonymous ftp: virtual.harvard.edu:/pub/htk/atm-forum/fcvc.ps.*

[13] Lyles, J., Swinehart, D., "The Emerging Gigabit Environment and the Role of Local ATM", *IEEE Communications Magazine*, April 1992.

[14] Lyles, B., lin, A., "Definition and Preliminary Simulation of a Rate-based Congestion Control Mechanism with Explicit Feedback of Bottleneck Rates", AF-TM 94-0708, July 1994.

[15] Newman, P., "Traffic Management for ATM Local Area Networks", *IEEE Communications Magazine*, August 1994.

[16] Pancha, P., El Zarki, M., "MPEG Coding for Variable Bit Rate Video Transmission", *IEEE Communications Magazine*, May 1994.

[17] "Ptolemy 0.5 User's Manual - Volume 1", College of Engineering, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1994.

[18] "Ptolemy 0.5 Star Atlas - Volume 2", College of Engineering, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1994.

[19] "Ptolemy 0.5 Programmer's Manual - Volume 3", College of Engineering, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1994.

[20] Schulzrinne, H., Kurose, J., Towsley, D., "An Evaluation of Scheduling Mechanisms for Providing Best-Effort, Real-Time Communication in Wide-Area Networks", *IEEE Proceedings of INFOCOM*, 1994.

[21] Turner, J.S., "New Directions in Communications (or which way to the information age?)", *IEEE Communications Magazine*, October, 1986.

[22] Woodruff, G.M., Rogers, R.G.H., Richards, P.S., "A Congestion Control Framework for High Speed Integrated Packetized Transport", *Proceedings of IEEE GLOBECOM*, November 1988.

[23] Zhang, L., "Virtual Clock: A New Traffic Control Algorithm for Packet-Switched Networks", *ACM Transactions on Computer Systems*, May 1991.

Appendix

A Rate Control Algorithms

Per-VC scheduler.

- Variables:
  1. *num* is found in the RM cell. It denotes the number of cells in a burst.
  2. t_{ideal} is the ideal rate.
  3. t_{want} is the desired rate.
  4. t_{iX} is the actual rate.
  5. *dif* is the number of late cells (accumulated).
  6. *cycle* is the number of cell slots between invocations of the EXPLICIT algorithm.
  7. *last* is a flag to be passed to the request procedure. It denotes whether the cycle is the last cycle in the current burst; i.e., ($last = 1$).

```
procedure Per-VC Scheduler:
begin
    { At every cycle time slots, for each $VC_i$ – }
    repeat:
    begin
        receive the RM cell and extract the number of cells in the
        burst, $num_i$, and the duration of the burst, $period_i$;
        $flag = FALSE$;
        $orig\_num_i = num_i$;
        $orig\_period_i = period_i$;
        $t_{ideal} = num_i/period_i$;
        $t_{want} = (num_i/period_i) + (dif/cycle)$;
        for i = 1 to $period_i/cycle$ do
        begin
            if $i == period_i/cycle$ then
                $last = 1$;
            else
                $last = 0$;
            $t_{iX}$ = request($t_{want}$, last);
            if $dif == 0$ then
                if $((ideal\_t == t_{iX})$ and $(i == 1))$ then
                    forward the same RM cell;
                    $flag = TRUE$;
                else if $((ideal\_t == t_{iX})$ and $NOT flag)$ then
                    forward a new RM cell with
                    $num_i = orig\_num_i - (i-1) \times (orig\_num_i/orig\_period_i) \times cycle$ and
                    $period_i = cycle \times ((orig\_period_i/cycle) - i + 1)$;
                    $flag = TRUE$;
                else if $(ideal\_t > t_{iX})$ then
```

```
                  forward a new RM cell with
                  ($num_i = t_{iX} \times cycle$) and ($period_i = cycle$);
                  $flag = FALSE$;
            else { $dif > 0$ }
                forward a new RM cell with
                $num_i = (t_{iX} \times cycle)$ and ($period_i = cycle$);
            forward data at rate $t_{iX}$ for $cycle$ slots;
            $dif = (t_{want} - t_{iX}) \times cycle$;
            if the next cell is an RM cell then break out of the for() loop;
            $t_{want} = ((ideal_t \times cycle) + dif)/cycle$;
        end { of for }
    end { of repeat }
end
```

Per-Output Port Rate Scheduling Algorithm.

- Variables:
  - Variables initialized/reset during call setup/teardown.
    - $GU[1..N]$ = array of guaranteed rates for VC_i's. Elements in this array are initialized during call setup per VC.
    - $FLAG[1..N]$ = array of flags which denote whether a VC is active/non-active. Elements in this array are set/unset during call setup/teardown per VC.
  - Variables passed in by the per-VC schedulers.
    - $WANT[1..N]$ = array of desired rates for VC_i's. Each element in this array is passed in (via request()) per VC.
    - $LAST[1..N]$ = array of flags which denote whether the current cycle is the last cycle in the frame ($LAST[] = 1$) or not ($LAST[] = 0$). Each element in this array is passed in (via request()) per VC.
  - Variables passed back to the per-VC schedulers.
    - $RATE[1..N]$ = array of assigned rates for VC_i's. These rates are assigned by the rate sharing algorithm. Each non-zero element is returned to the per-VC scheduler from which it is indexed.
  - Variables which are internal to the per-port rate scheduler.
    - $N\_GU[1..N]$ = array of non-guaranteed rates. $N\_GU[i]$ denotes the number of extra cells divided by the cycle length, or excess rate, VC_i would like to transmit in the next cycle if VC_i can only transmit $GU[i] * cycle$ cells in the current cycle; i.e., $N\_GU[i] = (WANT[i] - GU[i])$.
    - num is the number of active connections.
      item $ORDER_{late}[1..N]$ and $ORDER_{delay}[1..N]$ are arrays of 'relative' rates and VC identifiers to be prioritized i.e., sorted, in decreasing order of 'relative' rates. $ORDER_{late}$ corresponds to VCs in the *late* class, i.e., $LAST = 1$. $ORDER_{delay}$ corresponds to VCs in the *delayed* class, i.e., $LAST = 0$. Each element in $ORDER_x$ consists of a pair of values: $ORDER_x[].id$ and

$ORDER_x[].relrate$. $ORDER_x[].id$ is the VC identifier for the port scheduler. $ORDER_x[].relrate$ contains the 'relative' rate request of the VC, i.e., $ORDER_x[i].relrate = (WANT[i] - GU[i])/GU[i]$.

```
procedure Per-Output Port Rate Scheduling:
begin
    { The scheduler at each output port will resolve possible contention
        by invoking this procedure every cycle slots. }

    Initialize all elements in $ORDER_{late}[1..N]$ and $ORDER_{delay}[1..N]$ to 0;
    Initialize $WANT[1..N]$ and $LAST[1..N]$ according to requests from
        the per-VC schedulers; { Note that for any $i$, $WANT[i]$ may be
        non-zero ($FLAG[i] = TRUE$) even if $VC_i$ does not submit a
        new request or RM cell. }

    if $\sum_{FLAG[i]=TRUE} WANT[i] < 1$ then
        return $WANT[i]$ for all $i$ where $FLAG[i] = TRUE$;
        exit;
    else
        for all $i$ such that $FLAG[i] = TRUE$ do
            { compute $N\_GU[i]$ }
            if $WANT[i] > GU[i]$ then
                $N\_GU[i] = (WANT[i] - GU[i])$;
            else
                $N\_GU[i] = 0$;

        for all $i$ such that $FLAG[i] = TRUE$ do
            if $(LAST[i] == 0)$ then
                $++num\_delay$;
                $ORDER_{delay}[num\_delay].index = i$;
                $ORDER_{delay}[num\_delay].relrate = (WANT[i] - GU[i])/GU[i]$;
            else
                $++num\_late$;
                $ORDER_{late}[num\_delay].index = i$;
                $ORDER_{late}[num\_delay].relrate = (WANT[i] - GU[i])/GU[i]$;

        Sort $ORDER_{late}[1..N]$ in decreasing order with respect to
            $ORDER_{late}[].relrate$;
        Sort $ORDER_{delay}[1..N]$ in decreasing order with respect to
            $ORDER_{delay}[].relrate$;

        { distribute rates to VCs }
        $i = 0$;
        $left = 1 - \sum GU[i]$;
        for i = 1 to $N$ do
            if $WANT[i] < GU[i]$ then
                $left = left + (GU[i] - WANT[i])$;
        while $(num\_late > i)$ do
            $temp = ORDER_{late}[i].index$;
            if $left > N\_GU[temp]$ then
                return to $VC_{temp}$: $N\_GU[temp] + min(GU[temp], WANT[temp])$;
            $left = left - N\_GU[temp]$;
        else
            return to $VC_{temp}$: $left + GU[temp]$;
            exit the while loop;
        $++i$;
    if ($num\_late > i$) then
        while ($num\_late > i$) do
            $temp = ORDER_{late}[i].index$;
            return to $VC_{temp}$: $min(GU[temp], WANT[temp])$;
            $++i$;
        $i = 0$;
        while ($num\_delay > i$) do
            $temp = ORDER_{delay}[i].index$;
            return to $VC_{temp}$: $min(GU[temp], WANT[temp])$;
            $++i$;
            exit;

    { distribute rates to VCs which are in the delayed class. }
    $i = 0$;
    while ($num\_delay > i$) do
        $temp = ORDER_{delay}[i].index$;
        if $left > N\_GU[temp]$ then
            return to $VC_{temp}$: $N\_GU[temp] + min(GU[temp], WANT[temp])$;
            $left = left - N\_GU[temp]$;
        else
            return to $VC_{temp}$: $left + GU[temp]$;
            exit the while loop;
        $++i$;
    if ($num\_delay > i$) then
        while ($num\_delay > i$) do
            $temp = ORDER_{delay}[i].index$;
            return to $VC_{temp}$: $min(GU[temp], WANT[temp])$;
            $++i$;
end
```

B Buffer Control Algorithm

Per-Output Port Buffer Scheduling Algorithm.

- External variable and variable initialized during call setup.
    - $Tbuf_X$ is the number of buffer slots at output port X.
    - $g\_buf[1..N]$ = array of the guaranteed number of buffer slots for VC_is. This corresponds to a VC's burst tolerance.
- Variables which are internal to the per-port buffer scheduler.
    - *total* is the total number of buffer slots taken by incoming cells at port X.

- *count*[1..*N*]. Each element denotes the per-VC total number of buffer slots taken by incoming cells of a particular VC.
- $flag[1..N]$. $flag[i] = true$ implies that $count[i] > g\_buf[i]$. $flag[i] = false$ implies that $count[i] \leq g\_buf[i]$.
- *turn* denotes the VC, VC_{turn}, which must drop cells due to buffer overflow.

```
procedure Per-Output Port Buffer Scheduling:
begin
    { This scheduler executes at each output port, $X$. }

    Initialize both $total$ and $turn$ to equal 0;
    Initialize all elements in $flag[1..N]$ to FALSE;
    Initialize all elements in $count[1..N]$ to 0;

    { This procedure accepts incoming cells. }
    while (TRUE) do
    begin
        hold incoming cell from $VC_i$ in temporary buffer;
        $++total$;
        if ($total == Tbuf$) then
            while ($flag[turn] == FALSE$) do
                $turn = (turn + 1) \bmod N$;
            while (the last cell in $VC_{turn}$'s queue is not an RM cell) and ($flag[turn] == TRUE$) do
                discard the last cell from $VC_{turn}$'s queue;
                $--count[turn]$;
                if ($count[turn] \leq g\_buf[turn]$) then
                    $flag[turn] = FALSE$;
                $--total$;
            if ($flag[turn] == TRUE$) then
                discard the last cell (RM cell) from $VC_{turn}$'s queue;
                $--count[turn]$;
                $--total$;
            receive the new incoming cell;
            $++count[i]$;
        else
            receive the new incoming cell;
            $++count[i]$;
        if ($count[i] > g\_buf[i]$) then
            $flag[i] = TRUE$;
    end

    { This procedure releases cells. }
    while (TRUE) do
        release next cell from head of queue of $VC_i$;
        $--total$;
        $--count[i]$;
            if ($count[i] \leq g\_buf[i]$) then
                $flag[i] = FALSE$;
end
```

C Congestion Control Algorithm

- **Variables.**

  - c_flag_{iX} corresponds to a flag which indicates whether VC_i is experiencing congestion ($c\_flag_{iX} = true$) or no congestion ($c\_flag_{iX} = false$) at output port X.
  - max_rate denotes the rate threshold.
  - max_ql denotes the queue length threshold.
  - rep denotes the number of cycles for which everytime the congestion checking algorithm must be invoked.
  - $RATE[], flag[]$, and $GU[]$ are the same variables defined in the Per-Output Port Rate Scheduler Algorithm.

- **Congestion checking algorithm.**

```
procedure Congestion Checking:
begin
    { At every $cycle \times rep$ cell slots – }
    if ($\sum_i RATE[i] > max\_rate$) and ($\sum_i ql_i > max\_ql$) then
        for i = 1 to $N$ do
            $c\_flag_i = TRUE$;
            if $flag[i] = TRUE$ and $RATE[i] > GU[i]$ then
                send a backward congestion RM cell to $VC_i$'s upstream node;
    else if ($\sum_i RATE[i] < 0.7$) and ($\sum_i ql_i < low\_thresh$) then
        for i = 1 to $N$ do
            if $c\_flag_i = TRUE$ then
                send a backward no-congestion RM cell to $VC_i$'s upstream node;
                $c\_flag_i = FALSE$;
end
```

- **Rate Change Algorithm**

```
procedure Rate Changing:
begin
    receive an RM cell from the downstream node of $VC_i$;
    if RM cell is a backward congestion RM cell then
        $c\_flag_i = TRUE$;
        $RATE[i] = GU[i]$;
    if RM cell is a backward no-congestion RM cell then
        $c\_flag_i = FALSE$;
end
```