Fair Flow Control for ATM-ABR Multipoint Connections*

Sonia Fahmy, Raj Jain, Rohit Goyal, and Bobby Vandalore
Purdue University
Department of Computer Sciences
E-mail: fahmy@cs.purdue.edu
Raj Jain is now at jain@wustl.edu, http://www.cse.wustl.edu/~jain

Abstract: Multipoint-to-multipoint communication can be implemented by combining the point-to-multipoint and multipoint-to-point connection operations. In a multipoint-to-point connection, multiple sources send data to the same destination on a shared tree. Traffic from multiple branches is merged into a single stream after every merge point. It is impossible for the network to determine any source-specific characteristics since all sources in the multipoint connection may use the same connection identifiers. The challenge is to develop a *fair* rate allocation algorithm without per-source operations, as these are no longer equivalent to per-connection or per-flow operations.

We give fairness definitions for multipoint connections, and we design and simulate an O(1) fair ATM-ABR rate allocation scheme for point-to-point and multipoint connections. Simulation results show that the algorithm performs well and exhibits desirable properties. We discuss the main modifications necessary for any ATM-ABR rate allocation scheme to accommodate multiple sources.

1 Introduction

Multipoint communication is the exchange of information among multiple senders and multiple receivers. Multipoint support in Asynchronous Transfer Mode (ATM) networks is essential for efficient duplication and synchronization of data. Examples of multipoint applications include audio and video conferencing, and server and replicated database synchronization (see figure 1). Multipoint-to-point connections are especially important for overlaying Internet (IP) networks and simplifying end systems and edge devices [15]. In multipoint-to-point connections, only one connection needs to be set up even if there are multiple data sources.

*This research was sponsored in part by Rome Laboratory/C3BC Contract # F30602-96-C-0156. This paper and all our papers and contributions are available through http://www.cis.ohio-state.edu/~jain/

Figure 1: ATM multipoint communication

An efficient and flexible ATM multipoint service is a key factor to the success of ATM networks. Several issues need to be addressed in the ATM multipoint service definition, such as routing, signaling, and traffic management. In this paper, we focus on traffic management issues in the case of multiple sources. Specifically, we tackle the definition of fairness, and the *congestion and feedback control* problem for multipoint-to-point connections. The problem of consolidating control cells is not tackled in this paper (see [4] for an analysis of the solutions to feedback consolidation).

ATM networks currently offer two service categories for data traffic: the available bit rate (ABR) and the unspecified bit rate (UBR) services. Capacity left over by real-time traffic is fairly divided among the active ABR sources and indicated to the sources through closed-loop feedback control [5]. The most commonly adopted fairness definition is max-min fairness [7]. Intuitively, this means that all sources bottlenecked at the same node are allocated equal rates (or proportional rates using weights). This definition was developed for point-to-point connections, and in this paper, we extend it for multipoint connections, and discuss the development of a distributed algorithm to achieve fairness.

Multipoint-to-point ABR connections require feedback to be returned to the appropriate sources at the appropriate times. The bandwidth requirements for a virtual connection (VC) after a merge point is the sum of the bandwidths used by all sources whose traffic is merged (see figure 2). This is because the aggregate data rate after a merge point is the sum of all incoming data rates to the merge point [8]. Consolidating control cells in the forward direction is not necessary since the ratio of control cells to data cells after merging remains the same.

We have defined several types of fairness for multipoint-to-point VCs implemented as shared trees [3]. Among these, we believe that weighted *source-based* fairness is the most preferred because it is a simple and logical extension of point-to-point fairness definitions. To compute source-based fair allocations, a single N-to-one connection is treated as N one-to-one connections (in terms of bandwidth allocation), regardless of which VC each source belongs to.

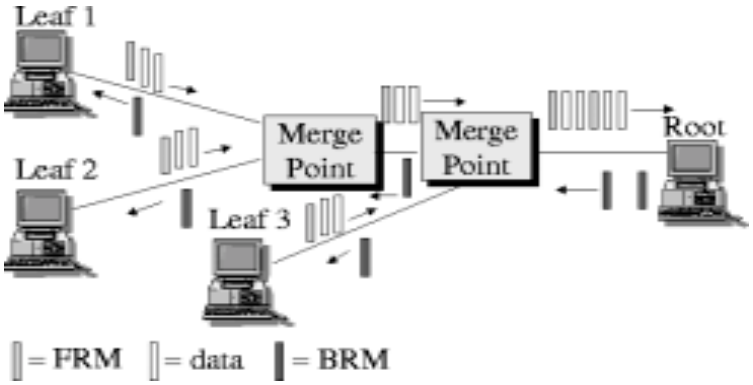


Figure 2: Multipoint-to-point connections

A source-based fair algorithm must give the same (or proportional) allocation to all sources bottlenecked on the same link. Source-based fairness in some switch implementations poses difficulties, since sources in the same VC cannot be distinguished (they have the same connection identifier). The challenges for rate allocation algorithms in this case include avoiding per-source accounting and avoiding estimating the number of active sources. This has to be done without adversely affecting the transient response or increasing the rate oscillations.

The remainder of this paper is organized as follows. First, we give some definitions and discuss the VC merge technique for avoiding cell interleaving in multipoint connections. Then, we summarize related work on multipoint-to-point algorithms. We define fairness using an example. In section 4 we develop the rate allocation and merge point algorithms for multipoint connections, and examine their features. We analyze the performance of the algorithm in section 5, and conclude with a set of recommendations for rate allocation schemes to support multiple sources.

2 Preliminaries and Related Work

In this section, we give some background on the problem of ABR multipoint flow control. We distinguish connections, sources and flows, discuss VC merging and ABR flow control, and discuss previous work on multipoint-to-point algorithms.

2.1 Connections, Sources and Flows

Definition: A component c_j is said to be *downstream* of another component c_i in a certain connection if c_j is on the path from c_i to the destination. In this case, c_i is said to be *upstream* of c_j. □

Figure 3 shows a configuration with two virtual connections (VCs). One of the VCs is a point-to-point VC, while the other is a multipoint-to-point VC. The sources in the multipoint-to-point

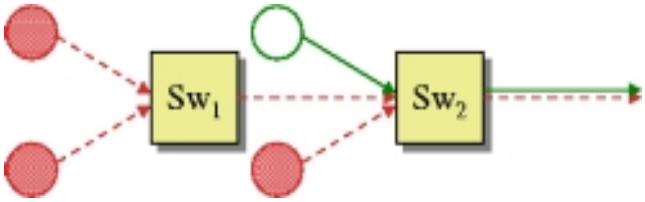


Figure 3: Source versus VC versus flow

VC are indicated by dark-colored circles, while the source in the point-to-point VC is denoted by the light-colored circle. At the second switch, traffic from four sources, but only two VCs, is being switched to the output port. Note, however, that the second switch can distinguish three input flows (the point-to-point connection, and two flows of the multipoint-to-point connection coming through different interfaces). The two sources whose traffic was merged at the first switch constitute a single flow at the second switch, assuming that they cannot be distinguished downstream of their merge point. Thus, two of the input flows that can be distinguished at the second switch belong to the same VC, while the third flow belongs to a different VC. The second switch merges the two flows of the same VC into a single flow.

2.2 VC Merging

In ATM networks, the virtual path identifier (VPI) and virtual connection identifier (VCI) fields in the cell header are used to switch ATM cells. The ATM adaptation layer (AAL) at the source segments packets into ATM cells, marking the last cell of each packet. The AAL at the destination uses the VPI/VCI fields and the end of packet marker to reassemble the data from the cells received.

ATM adaptation layer 5 (AAL5), which is used for most data traffic, does not introduce any multiplexing identifier or sequence number in ATM cells. If cells from different sources are merged and interleaved on the links of a multipoint connection (implemented as a shared tree), the AAL5 at the destination cannot assemble the data. This is because all traffic within the group uses the same VPI/VCI, and the identity of the source is not indicated in each cell. The AAL5 layer uses the end-of-message bit to determine the end of each packet, but since the cells of different packets are interleaved, all the packets may get corrupted, as illustrated in figure 4 (the end-of-message bit value is shown above each ATM cell in the figure).

One of the solutions proposed to this problem is the VC merge approach. This approach buffers the cells of packets coming through other interfaces at the switch until all cells of the current packet go through (see [6] for a description of the technique and [17] for an analysis of its performance). Thus, a packet-based scheduling algorithm is implemented at the merge point, and separate queues are maintained for each flow (where a flow is defined as the cells of a VC coming on an input

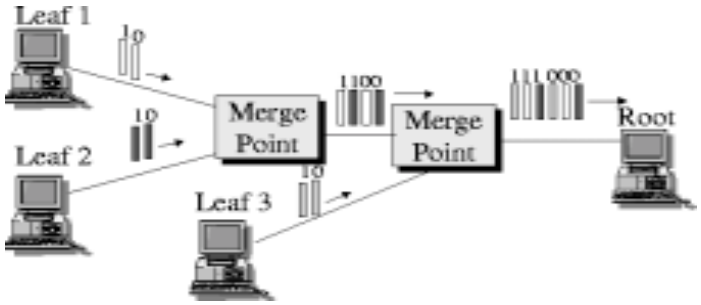


Figure 4: The cell interleaving problem

link). The end-of-message bit signals to the switch that a packet from a different port can now be forwarded. *In this paper, we focus on VC merge implementations, where cells from different sources are indistinguishable, since this is the most difficult case for bandwidth allocation algorithms to handle.*

2.3 ABR Flow Control

The available bit rate (ABR) service periodically indicates to sources the rate at which they should be transmitting. The feedback from the switches to the sources is indicated in resource management (RM) cells generated by the sources and turned around by the destinations (figure 5). The RM cells flowing from the source to the destination are called forward RM cells (FRMs) while those returning from the destination to the source are called backward RM cells (BRMs).

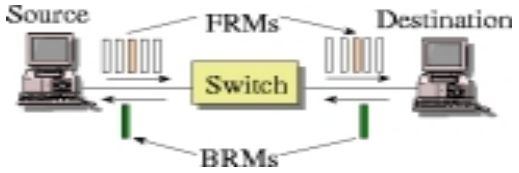


Figure 5: Resource management cells in an ATM network

The RM cells contain the source current cell rate (CCR), in addition to several fields that can be used by the switches to provide feedback to the sources. Feedback can be just one or two bits, or it can be the rate at which the source should transmit, called the explicit rate (ER). When a source receives a BRM cell, it computes its allowed cell rate (ACR) using its current ACR value, the congestion indication bits, and the explicit rate field of the RM cell.

2.4 Previously Proposed ABR Multipoint-to-Point Algorithms

Traffic management rules for *multipoint-to-point connections* are still in their early phases of definition [14, 13, 1, 12]. Ren and Siu [14, 13] have described an algorithm for multipoint-to-point congestion control, which assumes that VC merge is employed. The algorithm operates as follows.

A bit is maintained at the merge point for each of the flows being merged. The bit indicates that an FRM has been received from this flow after a BRM had been sent to it. Therefore, when an FRM is received at the merge point, it is forwarded to the root and the bit is set. When a BRM is received at the merge point, it is duplicated and sent to the branches that have their bit set, and then the bits are reset. We implement this algorithm as explained in section 4.2, and show simulation results in section 5.

In their papers [14] and [13], Ren and Siu only show simulation results for simple LAN configurations. We discuss more complex problems and many general algorithm design issues that arise in all multipoint algorithms, and show more simulation results for our proposed solutions. Furthermore, Ren and Siu's work does not clearly state which types of rate allocation algorithms the proposed multipoint extension works for. In fact, the extension does not work for many popular ABR schemes that perform per VC accounting, since this is no longer equivalent to per-source accounting.

Recently more complex algorithms have been developed [1, 12] for multipoint-to-point and multipoint-to-multipoint connections respectively. The algorithm in [1] aims at fairness among the sources as in [14]. The algorithm in [12] adds a *weight* in RM cells to allow scaling of the rates to give the appropriate allocations to sources. The throughput of a unicast source is given a pre-determined weight with respect to that of a sender in a multicast session. This technique adds more flexibility at the expense of complexity in RM cells and processing. Weight assignment is also very difficult.

3 Fairness Definition

Throughout the rest of the paper, we use max-min fairness as the underlying fairness definition. However, the definition we give apply for any underlying definition, e.g., general weighted fairness with minimum rate guarantees [16]. Max-min fairness means that no connection can be allocated a higher rate without hurting another connection having an equal or lower rate. We define a network *configuration* as a set of sources, destinations and switches, interconnected with links of given distances and bandwidths, and a set of virtual connections. We use the following notation:

n	denotes the number of sources in a given configuration
x_i	denotes the allocation given to the i^{th} source in a given configuration

Source-based. Source-based fairness allocates bandwidth fairly among all sources, regardless of which VC each source belongs to. Each N-to-one connection is treated the same as N one-to-one connections.

Definition: *Source-based* fairness divides bandwidth fairly among active sources as if they were sources in point-to-point connections, ignoring group memberships.
The allocation vector $\{x_1, x_2, \ldots, x_n\}$ is determined by applying the underlying fairness definition for all active sources x_i. □

Note, however, that the bandwidth allocated to a multipoint-to-point VC with N concurrent sources all bottlenecked on a certain link would be N times the bandwidth for a point-to-point VC bottlenecked on that same link, and N/K times that for a K-source multipoint-to-point VC bottlenecked on the same link.

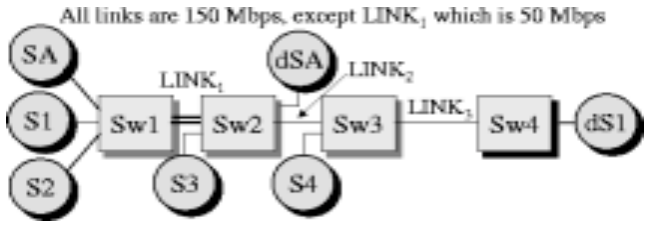


Figure 6: Example multipoint-to-point configuration with an upstream bottleneck

The fairness definition can be explained using the following example. Figure 6 illustrates a configuration with two VCs: one of the VCs is a multipoint-to-point VC with four sources and one destination, and the other is a point-to-point VC. Sources S_1, S_2, S_3 and S_4 are sending to destination dS_1, and source S_A is sending to destination dS_A. All links are approximately 150 Mbps (after SONET overhead is accounted for), except for the link between $Switch_1$ and $Switch_2$ ($LINK_1$) which is only 50 Mbps. Clearly, sources S_1, S_2 and S_A are bottlenecked at $LINK_1$, while sources S_3 and S_4 are bottlenecked at $LINK_3$. The aim of this example is to illustrate the allocation of the capacity left over by sources bottlenecked on $LINK_1$ to the sources bottlenecked on $LINK_3$.

The allocation vector according to the source-based definition is:

$\{S_1, S_2, S_3, S_4, S_A\} \leftarrow$ {16.67, 16.67, 58.33, 58.33, 16.67}

This is because each of sources S_1, S_2 and S_A is allocated one third of the bandwidth of $LINK_1$. At $LINK_3$, the $50 \times \frac{2}{3} = 33.33$ Mbps used by sources S_1 and S_2 is subtracted from the available bandwidth, and the remaining capacity (116.67 Mbps) is equally divided among sources S_3 and S_4.

Other fairness definitions, e.g., those in [3], are also possible. The choice of the type of fairness to adopt relies on the application type, and pricing methods used. We prefer *source-based* fairness, as it is a natural extension of point-to-point fairness definitions. In addition, it is the simplest to implement, and does not suffer from beat-down problems as with flow-based solutions. The interconnection unfairness discussed above is not a major problem since the number of concurrent senders in the multipoint connection is usually small in typical applications (e.g., one speaker at a time in

an audio conference). Weights can also be used to eliminate any unfairness. Pricing can be based on sources in this case. In the remainder of this paper, we discuss and analyze the development of an algorithm to achieve source-based fairness.

4 The Algorithm

We first discuss the rate allocation algorithm, and then the merge point algorithm. Then we discuss some design issues.

4.1 Rate Allocation Algorithm

Rate allocation algorithms are employed at every network switch to compute and indicate the appropriate feedback to the sources. The algorithm we discuss is based upon the ERICA+ rate allocation algorithm [10]. However, we eliminate all the steps that required per-VC accounting in ERICA+. The reason for this is that rate algorithms perform per-VC accounting as if it were *per-source* accounting. Per-source accounting must be avoided for compatibility with VC merge switches and for scalability.

The algorithm uses a measurement interval to measure the quantities required for computing the rate allocation. At the end of every interval, the algorithm averages some of the quantities measured, and uses these quantities to give the appropriate feedback to the sources in the following interval. The algorithm measures: (1) the ABR input rate to each port, and (2) the available capacity on each link, subtracting the capacity used by higher priority classes such as VBR. It also computes a function of the queueing delay and uses its value to scale the available capacity (in order to leave some of the capacity for the queues to drain). The ratio of the (average) measured input rate to the (average) mesaured target capacity is called the *overload factor.*

The algorithm also uses the current cell rate (CCR) of the sources, as indicated in the FRM cells. In addition, it keeps track of the maximum explicit rate indicated to all sources sending to this port during each interval.

The overload is compared to $1+\delta$ (usually δ is set to 0.1). If the overload is greater than 1.1, which means there is high overload, the algorithm scales down the current cell rate of the connection by the overload factor. Otherwise, if there is underload (overload is $\leq 1+\delta$), the algorithm also uses an additional quantity. This quantity is the maximum allocation allocated during the previous interval. Bringing up all allocations to this quantity ensures that all connections get fair rates according to the specified weights.

In the pseudocode below, there are two options that are not necessary for the algorithm, but help reduce rate fluctuations in some cases (especially when the measurement interval value is very small). The first option (which we label option 1) does not use the most current CCR value from FRM cells, but uses the maximum of the CCR values seen in FRMs in the current interval. This option is useful when there are multiple sources in the same VC, as explained in the next subsection. The second option (option 2) uses exponential averaging for the maximum ER given in the previous interval to smooth out variations.

The algorithm executes for each output port: when an FRM cell is received, when a BRM cell is received, and at the end of each measurement interval. The algorithm is O(1) and its complexity is independent of the number of connections and the number of sources. Since the calculations of the input rate, target capacity and overload factor are the same as in the ERICA+ algorithm, we only briefly outline these here.

FRM cell is received for VC j:

(current cell rate)$_j$ $\leftarrow$CCR field from the FRM cell

Or as an option (option 1: maximum CCR option):

IF (first FRM in interval)$_j$ = TRUE THEN

(current cell rate)$_j$ $\leftarrow$CCR field from the FRM cell

(first FRM in interval)$_j$ $\leftarrow$FALSE

ELSE

(current cell rate)$_j$ $\leftarrow$maximum (CCR field from the FRM cell, (current cell rate)$_j$)

END

BRM cell is to be sent out for VC j:

IF (overload factor $> 1+\delta$) THEN

ER $\leftarrow$(current cell rate)$_j$/overload factor

ELSE

ER $\leftarrow$maximum ((current cell rate)$_j$/overload factor, maximum ER in previous interval)

END

ER $\leftarrow$minimum (target capacity, ER)

maximum ER in current interval $\leftarrow$maximum (ER, maximum ER in current interval)

ER in BRM cell $\leftarrow$minimum (ER, ER in BRM cell)

End of measurement interval:

target capacity $\leftarrow$exponential average of (across intervals) of link capacity minus CBR and VBR

capacity, scaled for queues to drain by using a fractional function (refer to [10])

input rate ←exponential average (across intervals) of total ABR input cells being switched to this output port

overload factor ←input rate/target capacity

$\forall j$ (first FRM in interval)$_j$ ←TRUE

maximum ER in previous interval ←maximum ER in current interval

Or as an option (option 2: averaging the maximum ER in previous interval option):

maximum ER in previous interval ←

(1-α) $\times$ maximum ER in current interval + α $\times$ maximum ER in previous interval

maximum ER in current interval ←0

Notes:

1. The input rate, target capacity, overload factor, maximum ER in current interval and maximum ER in previous interval are computed and stored for each output port. The "first FRM in interval" (if used) and the "current cell rate" are stored for each VC for each output port.

2. In our simulations, the parameter δ is set to 0.1, and the parameter α is also set to 0.1. These are the recommended values for these parameters.

3. The "averaging of maximum ER in previous interval" option (option 2) slightly reduces rate oscillations in some cases. It is not essential if its implementation complexity is high.

4. The maximum CCR option (option 1) also reduces rate oscillations in cases of extremely small averaging interval values (< 200 μs for rates about 10 Mbps per source). It is also unnecessary. Exponentially averaging the maximum CCR values across intervals might further improve the performance. The next subsection discusses the usage of CCR in more detail.

Reference [10] gives a proof that this algorithm converges to the max-min fair rates for a single bottleneck case. The main idea of the proof is that the algorithm is fair because it allocates all sources bottlenecked at the same link the exact same rates. In addition, the algorithm converges to rates that result in an overload factor value close to one, because the rates are scaled by the overload factor.

4.2 Merge Point Algorithm

This algorithm is the same as the multipoint-to-point algorithm developed by Ren and Siu in [13]. The algorithm is employed at every merge point where cells from different sources in the same multipoint-to-point VC are being merged and follow the same path to the destination. We first give the pseudocode for the algorithm, and then discuss some properties of the algorithm.

A flag (can be one bit) called Ready is maintained for each of the *flows* being merged. The flag indicates that an FRM cell has been received from this flow after a BRM cell had been sent to it.

Upon the receipt of an FRM cell from branch i:

1. Forward FRM cell to the outgoing link
2. Let Ready$_i$ = TRUE

Upon the receipt of a BRM cell from the root:

```
FOR ALL upstream branches DO
    IF Ready_i = TRUE THEN
        Send a copy of the BRM to branch i
        Let Ready_i = FALSE
    END
END
```

When a BRM cell is about to be scheduled:

Perform the rate allocation algorithm as described in the previous section

Reference [13] gives a proof by induction on the number of levels of the multipoint tree to show that this algorithm gives fair allocations for multiple sources if the rate allocation algorithm employed gives max-min fair allocations.

4.3 Rate Allocation Design Issues

As previously mentioned, rate allocation algorithms for multipoint-to-point (or multipoint-to-multipoint) connections may not be able to distinguish cells from different sources in the same VC. Thus they cannot: (1) use the number of established connections as an indication of the number of sources, (2) measure or estimate the rate of each source, (3) distinguish between overloading and underloading sources, or compute the number of overloading sources, (4) estimate the effective number of active sources. Such techniques are used in many of the popular point-to-point switch schemes, such as the MIT scheme [2] and the UCSC scheme [9].

Most switch schemes also use the current cell rate of the sources in the computation of the explicit rate. **Algorithms which use the CCR values noted from backward RM cells are not fair for multipoint-to-point connections**. This is because it may be impossible to determine which source the RM cell belongs to. The CCR value in the BRM cells at the merge point may not capture upstream bottleneck information for *any* of the flows whose traffic is being merged, since it may actually be the CCR of a downstream source whose bottleneck rate is high. We explain this next.

Lemma 1: *Algorithms which use the CCR values noted from backward RM cells are not fair for multipoint-to-point connections:* $ER = f(CCR_{BRM})$ *is not necessarily max-min fair.*

Proof Sketch: The proof is by counter-example. We give a case where an algorithm using CCR_{BRM} gives unfair allocations. Suppose a multipoint-to-point VC has two sources, one of which has a bottleneck rate of 58 Mbps, and the other has a bottleneck rate of 16 Mbps, and the two sources are being merged at a switch. Figure 6 shows an example where at $Switch_2$, S_1 (and S_2) of rate 16 Mbps and S_3 of rate 58 Mbps are being merged (we will simulate this case in sections 5.2.2 to 5.2.4). The source which is bottlenecked at 16 Mbps (say S_1) shares its bottleneck link with a point-to-point connection (S_A to dS_A). At the merge point, BRM cells of the higher rate source (the 58 Mbps source) are more frequently sent to *all* the sources in this VC being merged with a high ER value (since the CCR is assumed to be 58 Mbps). This can result in over-allocation to the lower rate source(s) being merged, and unfairness to the point-to-point connection. □

Hence, algorithms that use the CCR value for rate computation must use the value of the CCR indicated in FRM cells for computation when a BRM cell is received. This is the most up-to-date value of CCR, since the CCR in the BRMs may be stale after traveling all the way to the destination and back. The CCR value in the FRM cells at the merge point captures upstream bottleneck information for one of the flows whose traffic is being merged. *The FRM cells of the sources being merged, however, may still be indistinguishable at the merge point. In the remainder of this section, we argue that this does not affect the convergence and steady state behavior of the algorithm.*

Lemma 2: *Algorithms which use the CCR values noted from forward RM cells can compute statistically fair allocations for multipoint-to-point connections.*

Proof Sketch:

Since the guaranteed fairness is statistical, the proof is also statistical. Assume that there are two flows S_{low} and S_{high} being merged. We will briefly examine the situation when the forward CCR used to compute the ER for a flow is not the CCR corresponding to that flow.

CASE 1:

When computing the ER for S_{low}, if the CCR of S_{high} is used, then the ER computed for S_{low} will

be too high. But S_{low} is bottlenecked upstream of the merge point (otherwise its bottleneck rate will not be less than that for S_{high}, since S_{low} and S_{high} merge at the merge point and never split after that), so the ER given to S_{low} at the merge point will be overwritten by upstream switches.

CASE 2:

For the case when the CCR of S_{low} is used to compute the ER for S_{high}, first consider the algorithm with the maximum CCR option. The only situation when the ER for S_{high} is calculated based upon the CCR for S_{low} is when only FRM cells of S_{low} have been seen since the beginning of the current interval. (Note that if no FRM cells have been seen at all, the CCR value used is the maximum seen in the previous interval, which will be the CCR of the higher rate source S_{high} unless S_{high} is sending at a very low rate, in which case the scheme should *not* allocate it high rates: see the discussion in [11] for more details on handling low rate sources.) Since S_{high} has a higher rate, it has a higher frequency of FRM cells, so it becomes highly improbable for this to hold.

This argument can be extended for the algorithm without the maximum CCR option. In this case, instead of the smaller CCR being used when only FRM cells from the lower rate source have been seen so far in this interval, it is the last FRM cell received that determines the CCR used. But, again, since the higher rate source has a higher FRM rate, it is statistically unlikely for the smaller CCR to be used. The maximum ER in the previous interval term ensures that if the small CCR is in fact used, the source is allocated at least as much as other VCs going to the same output port, which ensures fairness and fast convergence. □

4.4 Merge Point Design Issues

There are a number of ways to implement multipoint-to-point merge point algorithms. Each method offers a tradeoff in complexity, scalability, overhead, response time and steady state behavior.

In the above algorithm, a BRM cell is returned to a source for every one or more FRM cells it sends. Thus the BRM to FRM cell ratio at the source is less than or equal to one. In steady state, the ratio is likely to approach one, since the FRM rate and BRM rate will be similar. This is an important property of ABR flow control that should be maintained for multipoint-to-point connections. The BRM to FRM ratio *in the network* is also one in this case. (If FRM cells are turned around at merge points as in [14], the same FRMs can be turned around at another merge point or the destination, creating BRM cells that eventually get discarded in the network.)

Also observe that in this scheme, since the merge point does not need to turn around every FRM cell, the overhead of the algorithm is reduced. However, the scheme needs to duplicate BRM cells. With the new advances in multicast ATM switch architectures, this operation can be quite efficient.

The algorithm we use returns a BRM cell received from the root to the branches which have sent FRM cells to the merge point since the last BRM cell had been passed. This makes the scheme less sensitive to the number of levels of merge points, as compared to those schemes which turn around FRM cells (such as the scheme in [14]). This is because schemes turning around FRMs have to wait for an FRM to be received at every merge point, so their response time increases with the number of levels in the tree. In addition, the ER value returned by such schemes may be incorrect if no BRM cells have been received since the last one was sent, leading to rate oscillations and possibly large queue lengths.

5 Performance Analysis

This section provides a simulation analysis of the multipoint algorithm described in the previous two sections. Only a few simple experiments are shown here; more stringent tests have been conducted, and the preliminary results are consistent with those presented next.

The results are presented in the form of four graphs for each configuration:

(a) Graph of allowed cell rate (ACR) in Mbps versus time for each source

(b) Graph of ABR queue lengths in cells versus time at the bottleneck port of each switch

(c) Graph of link utilization versus time for each of the main (backbone) links (those that connect two switches to each other)

(d) Graph of number of cells received versus time for each destination

5.1 Parameter Settings

Throughout our experiments, the following parameter values are used:

1. Except where otherwise indicated (in sections 5.2.2 to 5.2.4), all links have a bandwidth of 155.52 Mbps (149.76 Mbps after SONET overhead is accounted for).

2. All multipoint-to-point traffic flows from the leaves to the root of the tree. No traffic flows from the root to the leaves, except for RM cells. Point-to-point connections are also unidirectional.

3. Except in section 5.2.3 where we experiment with the source parameter rate increase factor (RIF), we have set RIF to 1/32 in our simulations. We do not, however, expect the performance of the algorithm to be significantly influenced by the value of RIF, as seen in section 5.2.3.

4. The source parameter transient buffer exposure (TBE) is set to large values to prevent rate decreases due to the triggering of the source open-loop congestion control mechanism. This was done to isolate the rate reductions due to the switch congestion control scheme from the rate reductions due to TBE.

5. All other ABR parameters are set to their default values [5].

6. A dynamic queue control function is used to scale the available capacity and achieve a constant queuing delay in steady state [10]. The "target delay" parameter specifies the desired queuing delay. A value of 1.5 ms was used. An inverse hyperbolic function is used. The hyperbolic function curve parameters used were $a = 1.15$ and $b = 1$. The queue drain limit factor is set to 0.5 (which means that up to 50% of the link capacity can be used to drain queues).

7. A fixed time measurement interval is used to measure and average the input rate and available capacity, and to note the maximum allocation given (and possibly the maximum CCR value in FRM cells). The interval is set to 5 ms in all experiments except those in section 5.2.4.

8. Since we do not implement VC merge in our switches, we only use one cell long packets. Our next study will implement VC merge and examine its effect.

9. All sources are deterministic, i.e., their start/stop times and their transmission rates are known.

10. Simulation time is two seconds.

11. The simulations use both the maximum CCR option and exponentially averaging the maximum ER option as discussed in section 4.1. We have simulated all our configurations without using either option, and with each option separately, and the differences were insignificant. We do not show these results here for space considerations. In particular, the results when neither of the two options is enabled, and with extremely small measurement intervals (as with the simulations in section 5.2.4) showed that the algorithm still rapidly converges to the optimal allocations, and that the oscillations (though they do slightly increase) were *not* significantly more than the results we show in section 5.2.4.

5.2 Simulation Results

In this section, we discuss a *sample* of our simulation results. We mainly use two configurations, and experiment with different link lengths, initial cell rates of the sources, rate increase factor values, and lengths of the measurement interval.

5.2.1 Downstream Bottleneck Configuration

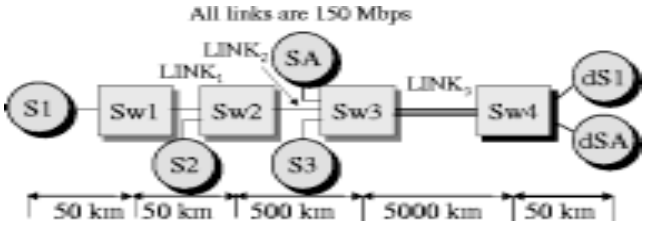


Figure 7: Example multipoint-to-point configuration with a downstream bottleneck

Figure 7 illustrates a configuration with two VCs: one of the VCs is a multipoint-to-point VC with three sources and one destination, and the other is a point-to-point VC. Sources S_1, S_2, and S_3 are sending to destination dS_1, and source S_A is sending to destination dS_A. All links are 149.76 Mbps (OC-3 links after SONET overhead is accounted for), and their lengths are as shown in the figure. Clearly, all four sources are sharing a bottleneck link ($LINK_3$) between $Switch_3$ and $Switch_4$. This experiment shows the division of the capacity of this bottleneck link among all sources in both types of connections.

Applying the fairness definition among sources, the optimal allocations should be:

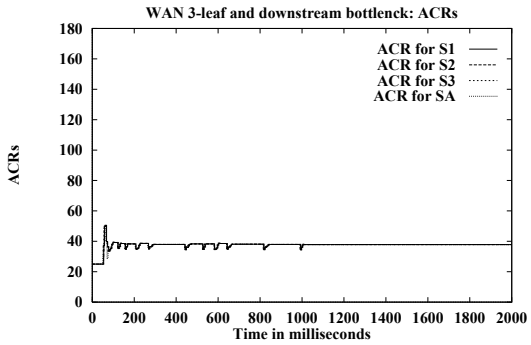
$\{S_1, S_2, S_3, S_A\} \leftarrow \{37.5, 37.5, 37.5, 37.5\}$

Each of the four sources is allocated $\frac{1}{4} \times 150 = 37.5$.

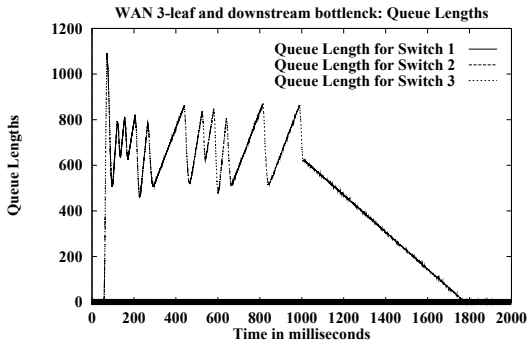
Figure 8 illustrates the results of simulating the above configuration. The sources start with an ICR value of 25 Mbps, which is below their optimal allocation. Clearly, all sources rise to their optimal rates quickly (figure 8(a)), and the queues are small (figure 8(b)). The bottleneck link ($LINK_3$) is fully utilized (figure 8(c)). $LINK_2$ is 50% utilized (since only 2 of the 4 sources utilize it) and $LINK_1$ is only 25% utilized (1 out of 4 sources).

Observe that with source-based fairness, VCs that have a larger number of concurrently active sources get more bandwidth than VCs with less concurrent sources on the same link. This can be clearly seen from the slope of the cells received graph (figure 8(d)) for dS_1 and dS_A. Clearly dS_1 has a slope that is three times as large as that for dS_A. After 2 seconds, the ratio of cells received at dS_A to dS_1 is around 175000 to 520000, which is exactly 1 to 3. Thus the resource allocation is not fair among the VCs.

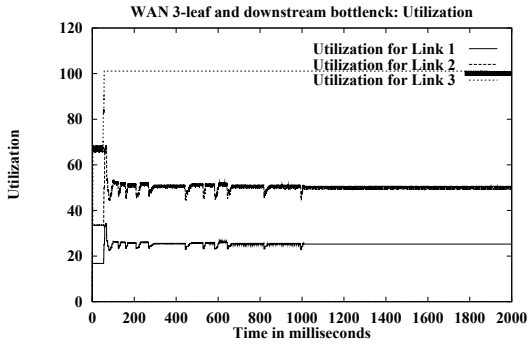
Figure 9 illustrates the results of simulating the same configuration (figure 7) when all sources start at a high ICR value. The ICR for all sources here is 100 Mbps. This creates an initial overload on $LINK_3$ of $\frac{400}{150} = 2\frac{2}{3}$. The algorithm recovers from this situation and all sources converge to the correct value of approximately 37.5 Mbps (figure 9(a)). The queues at $Switch_3$ start dropping after approximately one round trip (figure 9(b)).



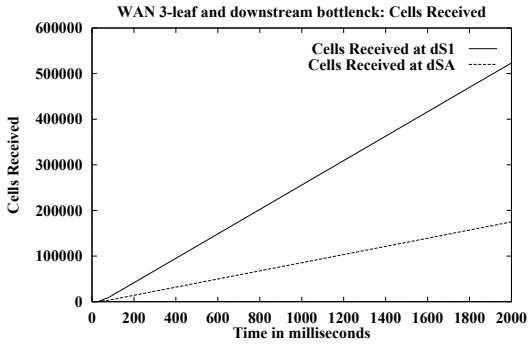
(a) Allowed Cell Rate



(b) Queue Length

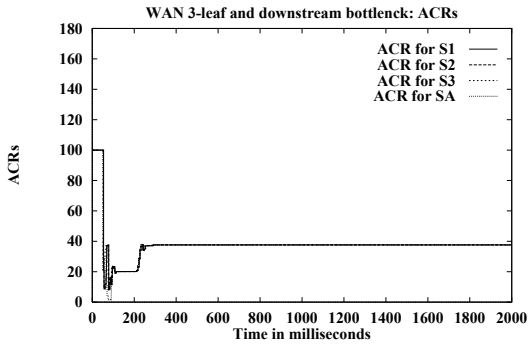


(c) Link Utilization

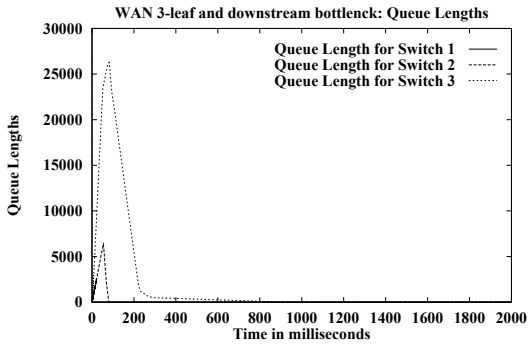


(d) Cells Received

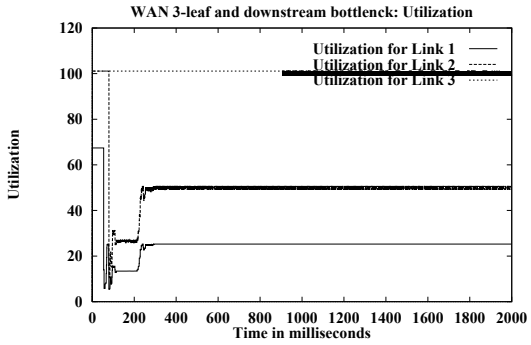
Figure 8: Results for a WAN multipoint-to-point configuration with a downstream bottleneck (long LINK3, low ICR)



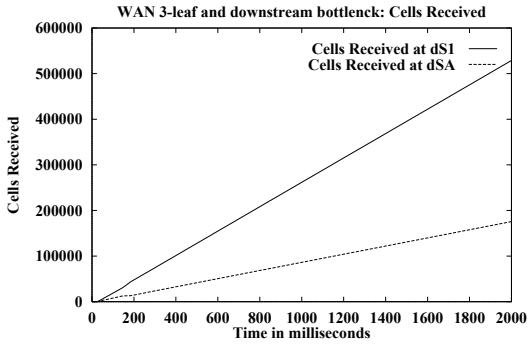
(a) Allowed Cell Rate



(b) Queue Length

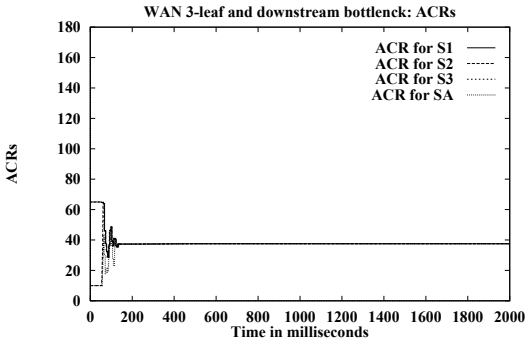


(c) Link Utilization

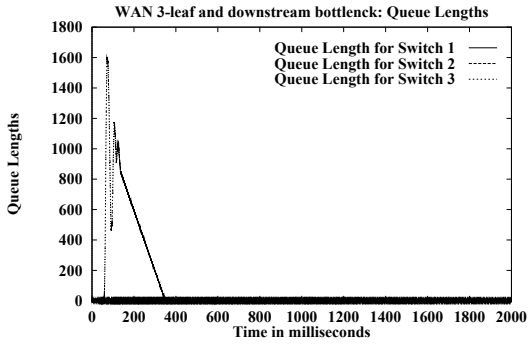


(d) Cells Received

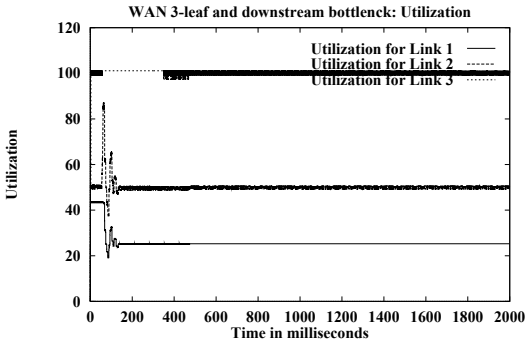
Figure 9: Results for a WAN multipoint-to-point configuration with a downstream bottleneck (long LINK3, high ICR)



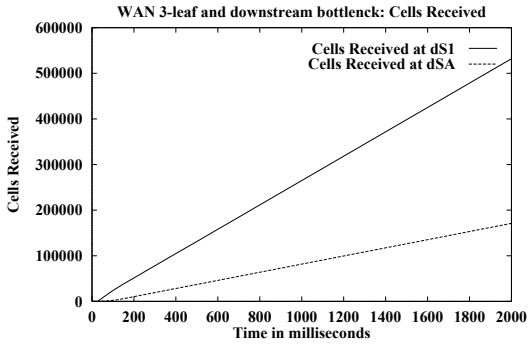
(a) Allowed Cell Rate



(b) Queue Length



(c) Link Utilization



(d) Cells Received

Figure 10: Results for a WAN multipoint-to-point configuration with a downstream bottleneck (long LINK3, different ICRs)

Figure 10 shows the results for the same configuration, but different sources start at different ICR values. Sources S_1 and S_3 start at an ICR of 65 Mbps, while sources S_2 and S_A start at 10 Mbps. Notice that the sum of the source rates for all sources is 150 Mbps, so the initial load value is close to 1. The rates for sources S_1 and S_3 are quickly reduced, while those of sources S_2 and S_A quickly rise, as seen in figure 10(a). The queues are also quite small (figure 10(b)).

5.2.2 Upstream Bottleneck with Heterogenous Links Configuration

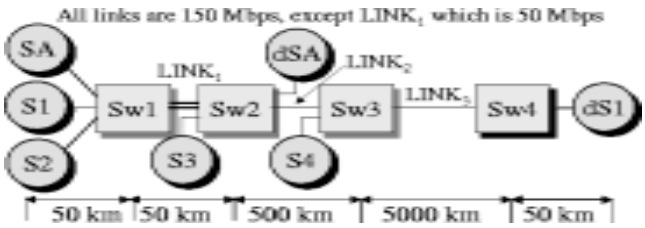


Figure 11: Example multipoint-to-point configuration with an upstream bottleneck

Figure 11 illustrates the same configuration as in figure 6, where all links are approximately 150 Mbps, except for the link between $Switch_1$ and $Switch_2$ ($LINK_1$) which is **only 50 Mbps**. The link lengths are as shown in the figure.

Recall that the allocation vector according to the source-based fairness definition is:

$\{S_1, S_2, S_3, S_4, S_A\} \leftarrow \{16.67,\ 16.67,\ 58.33,\ 58.33,\ 16.67\}$

Figure 12 illustrates the results for this configuration. Sources S_1 and S_2 start at an ICR of 20 Mbps. Source S_3 starts at 30 Mbps and source S_4 starts at 80 Mbps. Source S_A starts at 10 Mbps.

As seen in figure 12(a), sources S_1, S_2 and S_A converge to about 16.67 Mbps, while sources S_3 and S_4 converge to about 58.33 Mbps. The queues are bounded to reasonable values (figure 12(b)) and utilization of the bottleneck links ($LINK_1$ and $LINK_3$) are close to 100% (figure 12(c)). Destination dS_A gets much less throughput than dS_1 (figure 12(d)), since source S_A is bottlenecked on a 50 Mbps link with 2 other sources. After 2 seconds, the ratio of the throughputs for destinations dS_A to dS_1 is approximately 80000 to 700000 which is 0.11. The slopes of the two lines also have the same ratio. This is close to the optimal value since $16.67/149.76 = 0.11$.

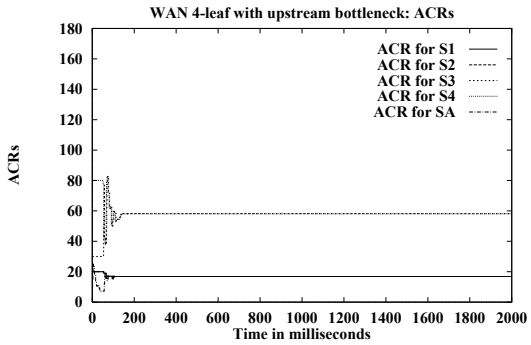
5.2.3 Effect of Large Rate Increase Factor Values

The rate increase factor determines the maximum increase when a BRM cell indicating underload is received. If the RIF is set to a fraction less than one, the maximum increase at each step is limited to RIF $\times$ the peak cell rate for the VC. Setting RIF to small values is a more conservative strategy that controls queue growth and oscillations, especially during transient periods. It, however, may slow down the response of the system when capacity suddenly becomes available leading to underutilization.

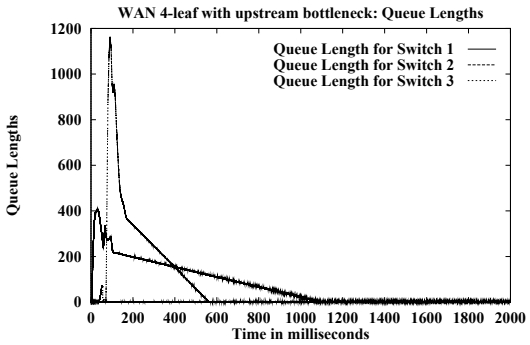
Figure 13 illustrates the results for the configuration of figure 11 when the rate increase factor (RIF) is set to its maximum possible value, which is 1. Part (a) of the figure shows that the rates do not oscillate more than the corresponding figure with a small RIF value (figure 12(a)). The queues in figure 13(b) are also similar to those in figure 12(b).

5.2.4 Effect of Extremely Short Measurement Intervals

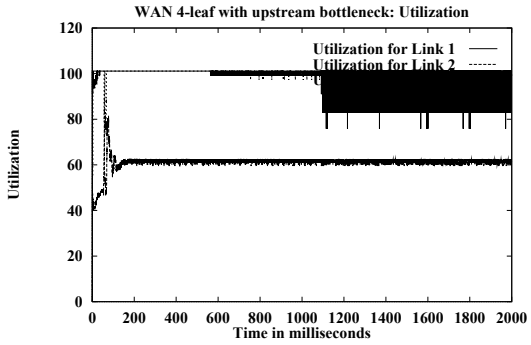
As discussed in section 4.1, extremely short measurement intervals can cause the algorithm to suffer from oscillations. To examine this effect, we have simulated the algorithm with a measurement interval of 200 μs. Recall that in the upstream bottleneck configuration (shown in figure 11), the optimal rates for sources S_1, S_2 and S_A are 16.67 Mbps, and those for sources S_3 and S_4 are



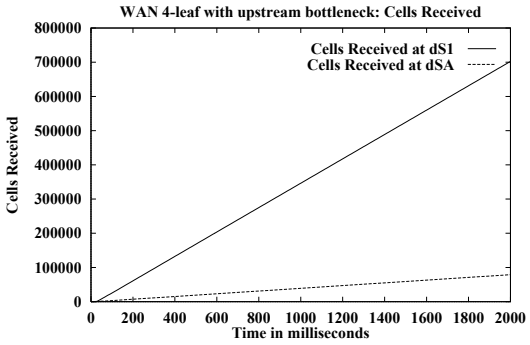
(a) Allowed Cell Rate



(b) Queue Length

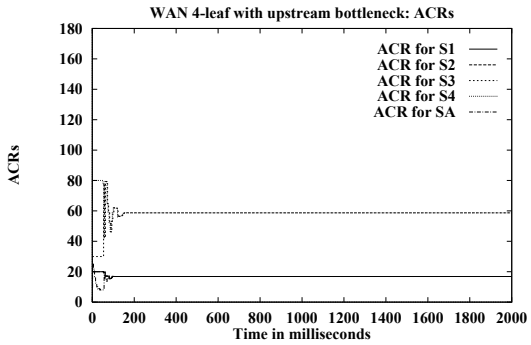


(c) Link Utilization

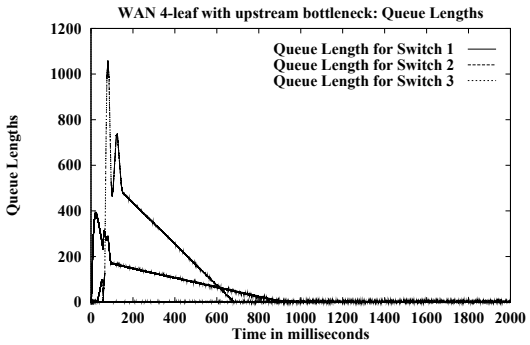


(d) Cells Received

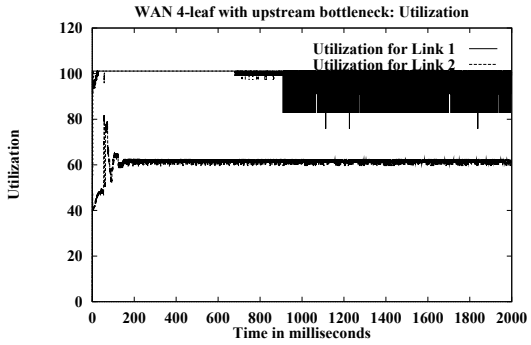
Figure 12: Results for a WAN multipoint-to-point configuration with an upstream bottleneck (long LINK3)



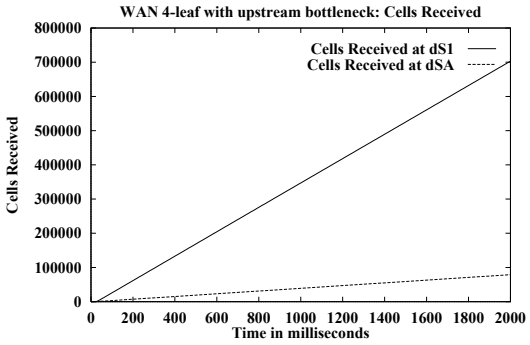
(a) Allowed Cell Rate



(b) Queue Length



(c) Link Utilization



(d) Cells Received

Figure 13: Results for a WAN multipoint-to-point configuration with an upstream bottleneck (long LINK3, large RIF)

58.33 Mbps. This implies that, in steady state, RM cells for sources S_1, S_2 and S_A arrive every:

$$\frac{Nrm \times bits/cell}{ACR} = \frac{32 \times 53 \times 8}{16.67\ M} = 813.92\ \mu s$$

For sources S_3 and S_4, RM cells arrive every:

$$\frac{Nrm \times bits/cell}{ACR} = \frac{32 \times 53 \times 8}{58.33\ M} = 232.61\ \mu s$$

where a source sends an FRM cell every Nrm cells, and the default value of Nrm is 32.

Setting the measurement interval to 200 μs means that RM cells for S_3 and S_4 might not be received every measurement interval, and that RM cells for S_1, S_2 and S_A might not be received for 4 consecutive measurement intervals.

In order to receive at least one FRM cell from the highest rate source in a certain interval, the interval length should be $> \frac{Nrm}{ACR\ maximum}$. This condition is likely to hold for reasonably long intervals, unless *all* sources are sending at very low rates, in which case the overload factor will be low and their rates will increase if they have data to send.

Figure 14 illustrates the results for the configuration of figure 11. Clearly, the short averaging interval causes more oscillations, but the rates of the sources still converge to their fair rates. Also observe that the number of cells received for both connections is the same as in figure 12(d). Increasing the value of the parameter α (in section 4.1) can reduce the oscillations.

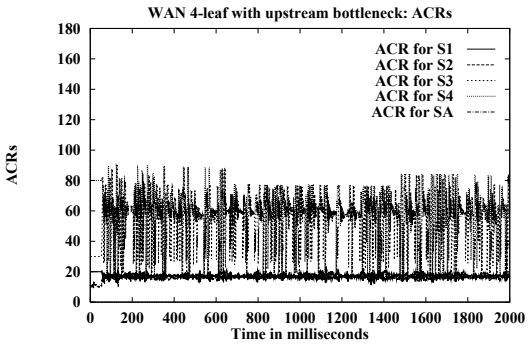
6 Conclusions and Recommendations for Switch Schemes

All source-based switch algorithms operating in VC merge switches need to avoid distinguishing among sources in the same VC. Key lessons learned from this study include (refer to section 4.3 for supporting arguments):

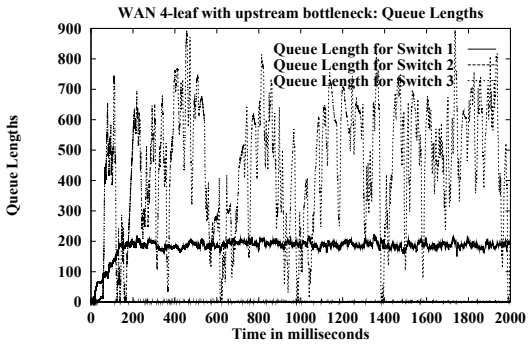
1. Source-level accounting should not be performed in multipoint rate allocation algorithms. For example, measuring the rates for each source, or distinguishing overloading and underloading sources cannot be performed. If such accounting is performed at the VC level or the flow level, an additional mechanism to divide VC or flow bandwidth among sources is necessary.

2. Estimating the effective number of active sources in order to divide the available capacity among them is very difficult in multipoint connections, since it is impossible to distinguish among sources in the same multipoint VC with VC merge implementations.

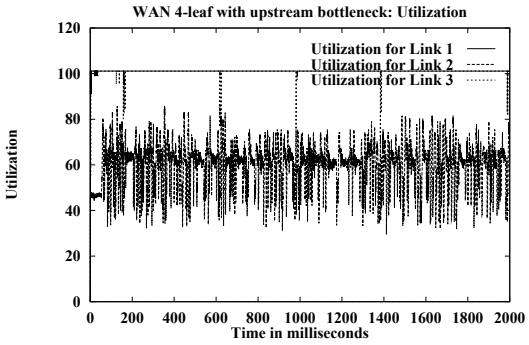
3. The only information a multipoint rate allocation algorithm can use is the information supplied in RM cells, in addition to *aggregate* measurements of load, capacity and queuing delays.



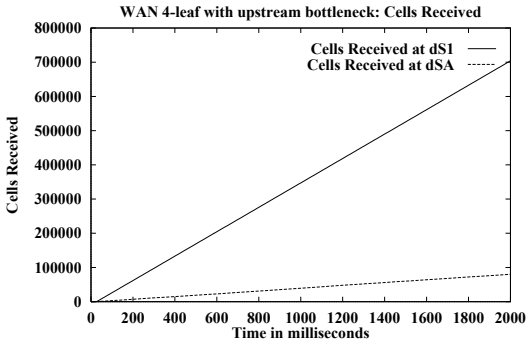
(a) Allowed Cell Rate



(b) Queue Length



(c) Link Utilization



(d) Cells Received

Figure 14: Results for a WAN multipoint-to-point configuration with an upstream bottleneck (long LINK3, short interval)

4. CCR values from BRM cells should not be used in computing rate allocations for sources in multipoint connections, since the CCR value can be that of another source that does not go through the switch performing the computation. That source may have a much higher bottleneck rate, and using its CCR can result in unfairness.

5. CCR values from FRM cells can be used to compute rate allocations for sources in multipoint connections, even though the CCR used to compute the rate for a source may not actually be the CCR value of the source. This does not create problems due to the properties of the merged flow (see section 4.1 for a more detailed explanation). The maximum CCR value seen in an interval can be used instead of the CCR of the source. Exponential averaging of the maximum CCR seen or maximum ER given may further improve the performance of the algorithm.

6. Merge point algorithms should avoid changing the BRM to FRM ratio at the source or inside the network, to maintain the rate of feedback that the source requires, and avoid excessive overhead in the network. Scalability of the scheme is also affected by these ratios. Excessive complexity, noise, and response time can also be avoided by returning the BRM cells coming from the root, instead of turning around the RM cells at the merge points (refer to section 4.4 for supporting arguments).

We have given and simulated an O(1) algorithm for computing source-based fair allocations for multipoint-to-point and point-to-point connections. The algorithm uses simple aggregate measurements and maximum CCR values from FRM cells during successive intervals to perform rate computation. The algorithm exhibited very good behavior for the configurations tested. More extensive performance analysis is crucial to examine the fairness, complexity, overhead, transient response, delays, and scalability tradeoffs in multipoint algorithm design. Extending multipoint-to-point schemes for multipoint-to-multipoint connections can be performed by combining point-to-multipoint algorithms (such as those developed in [4]) with the multipoint-to-point algorithm.

References

[1] D. Cavendish and M. Gerla. Rate based congestion control for many-to-many multicast ABR traffic. In *Proceedings of SPIE '98 Conference on Performance and Control of network systems II*, volume 3530, pages 122–130, November 1998.

[2] A. Charny, D. Clark, and R. Jain. Congestion Control with Explicit Rate Indication. In *Proceedings of ICC*, June 1995.

[3] S. Fahmy, R. Jain, R. Goyal, and B. Vandalore. Fairness for ABR multipoint-to-point connections. In *Proceedings of SPIE '98 Conference on Performance and Control of network systems II*, volume 3530, pages 131–142, November 1998.

[4] Sonia Fahmy, Raj Jain, Rohit Goyal, Bobby Vandalore, Shivkumar Kalyanaraman, Sastri Kota, and Pradeep Samudra. Feedback consolidation algorithms for ABR point-to-multipoint connections. In *Proceedings of the IEEE INFOCOM*, volume 3, pages 1004–1013, March 1998.

[5] The ATM Forum. The ATM forum traffic management specification version 4.0. ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.000.ps, April 1996.

[6] Matthias Grossglauser and K. K. Ramakrishnan. SEAM: Scalable and efficient ATM multipoint-to-multipoint multicasting. In *Proceedings of the IEEE INFOCOM*, April 1997.

[7] Jeffrey M. Jaffe. Bottleneck flow control. *IEEE Transactions on Communications*, COM-29(7):954–962, July 1981.

[8] Mark Jeffrey. Scope, concepts and issues for the new multiway BOF. ATM Forum/96-0628, June 1996.

[9] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. An efficient rate allocation algorithm for ATM networks providing max-min fairness. In *Proceedings of the 6th IFIP International Conference on High Performance Networking*, September 1995.

[10] S. Kalyanaraman, Raj Jain, Sonia Fahmy, Rohit Goyal, and Bobby Vandalore. The ERICA switch algorithm for ABR traffic management in ATM networks. *IEEE/ACM Transactions on Networking*, 1998. Submitted. Available through our web page.

[11] Shivkumar Kalyanaraman, Raj Jain, Rohit Goyal, Sonia Fahmy, and Seong-Cheol Kim. Use-it or lose-it policies for the available bit rate (ABR) service in ATM networks. *Journal of Computer Networks and ISDN Systems*, 1998.

[12] W. M. Moh and Y. Chen. Design and evaluation of multipoint-to-point multicast flow control. In *Proceedings of SPIE '98 Conference on Performance and Control of network systems II*, volume 3530, pages 143–154, November 1998.

[13] W. Ren, K-Y Siu, H. Suzuki, and M. Shinohara. Multipoint-to-multipoint ABR service in ATM. *Journal of Computer Networks and ISDN Systems*, 30(19):1793–1810, October 1998.

[14] Wenge Ren, Kai-Yeung Siu, and Hiroshi Suzuki. Multipoint-to-point ABR service in ATM networks. In *Proceedings of the International Conference on Communications, ICC'97, Montreal*, June 1997.

[15] E. M. Spiegel. Multipoint connection support in PNNI 2.0. *ATM Forum Perspectives, IEEE Network*, Nov/Dec 1997.

[16] B. Vandalore, S. Fahmy, R. Jain, R. Goyal, and M. Goyal. A definition of general weighted fairness and its support in explicit rate switch algorithms. In *Proceedings of the Sixth International Conference on Network Protocols 1998 (ICNP '98)*, pages 22–30, October 1998. http://www.cis.ohio-state.edu/~jain/papers/icnp98_bv.htm.

[17] Indra Widjaja and Anwar I. Elwalid. Performance issues in VC-merge capable switches for IP over ATM networks. In *Proceedings of the IEEE INFOCOM*, volume 1, pages 372–380, March 1998.

All our papers and ATM Forum contributions are available through http://www.cis.ohio-state.edu/~jain/