****

ATM Forum Document Number: ATM Forum/97-0832

****

Title: Fairness for ABR multipoint-to-point connections

****

Abstract:
Fair bandwidth management for multipoint-to-point ABR connections is an extremely important problem. In multipoint-to-point connections, the traffic at the root (destination) is the sum of all traffic originating at the leaves. The most crucial concern in the case of multiple senders is how to define fairness within a multipoint group and among multipoint groups and point-to-point connections [3]. This can be complicated since the multipoint connection can have the same identifier (VPI/VCI) on each link, and senders might not be distinguishable in this case. We give various possibilities for defining fairness, and show the tradeoffs involved. In addition, we show that switch algorithms need to be adapted to give fair allocations for multipoint-to-point connections. This is because many rate allocation algorithms implicitly assume that there is only one sender in each VC, which does not hold for multipoint-to-point cases.

****

Source:
Sonia Fahmy, Raj Jain, Rohit Goyal, and Bobby Vandalore
The Ohio State University (and NASA)

Raj Jain is now at Washington University in Saint Louis, jain@cse.wustl.edu http://www.cse.wustl.edu/~jain/ cis.ohio-state.edu

Sastri Kota
Lockheed Martin Telecommunications
1272 Borregas Avenue
Bldg B/551 O/GB - 70
Sunnyvale, CA 94089
E-mail: sastri.kota@lmco.com

Pradeep Samudra
Director, Systems Engineering
Broadband Network Lab
Samsung Telecom America, Inc.
1130 E Arapaho
Richardson, TX 75081

Phone: 972-761-7865

E-mail: psamudra@telecom.sna.samsung.com

****

Date: September 1997

****

Distribution: ATM Forum Technical Working Group Members (AF-TM)

****

Notice:

This contribution has been prepared to assist the ATM Forum. It is offered to the Forum as a basis for discussion and is not a binding proposal on the part of any of the contributing organizations. The statements are subject to change in form and content after further study. Specifically, the contributors reserve the right to add to, amend or modify the statements contained herein.

****

1 Introduction

Switch rate allocation algorithms generally service CBR and VBR traffic in preference to ABR traffic. The left-over capacity is *fairly* divided among the active ABR sources [5]. The most commonly adopted fairness definition is max-min fairness [10]. Intuitively, this means that all sources bottlenecked at the same node are allocated equal rates. This definition was developed for point-to-point connections, and in this contribution, we attempt to extend it for multipoint connections.

For point-to-multipoint ABR connections, the source is controlled to the minimum rate supported by all the leaves of the multipoint tree. Therefore, the extension of the max-min fairness definition is straightforward. ***With multipoint-to-point and multipoint-to-multipoint connections, however, the implicit assumption that each connection has only one source is no longer valid.***

In this contribution, we define several methods for computing the max-min fair allocations for multipoint-to-point VCs, and discuss the necessary modifications to switch schemes to give these allocations.

The remainder of this contribution is organized as follows. The next section discusses the solutions to the merging and cell interleaving problem for multipoint connections. Then, previous work in multipoint-to-point algorithms is summarized. We present our max-min fairness definitions in section 4, and show their operation, merits and drawbacks with the aid of an example. We then discuss several design issues (section 5), and show how switch schemes need to be adapted to give max-min fair allocations in section 6. The contribution concludes with a summary of the tradeoffs and issues involved.

2 Cell Interleaving Solutions

In ATM networks, the Virtual Path Identifier (VPI)/Virtual Connection Identifier (VCI) fields in the cell header are used to switch ATM cells. The ATM adaptation layer (AAL) at the sender segments packets into ATM cells, marking the last cell of each packet. The AAL at the receiver 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 most commonly used with data traffic, does not contain any multiplexing identifier or sequence number. If cells from different senders are interleaved on the links of a multipoint connection (implemented as a shared tree), the AAL5 at the receiver cannot assemble the data. This is because all traffic within the group uses the same VC identifier (see figure 1). 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 are corrupted. The identity of the
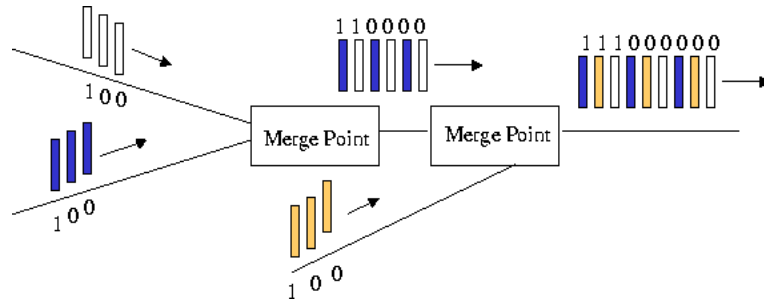
Figure 1: The cell interleaving problem

sender is not indicated in each cell. Hence, alternate solutions must be implemented.

Solutions to this problem attempt to either entirely avoid merging, or to prevent interleaving of cells of packets originating from different sources on the same multipoint connection after merging, or to provide enough information in the cell headers to enable the receivers to reassemble the packets even if their cells are interleaved.

The solutions proposed to the cell interleaving problem include:

1. **AAL3/4:** AAL3/4 can be used instead of AAL5. AAL3/4 contains a 10-bit multiplexing identifier (MID) field, part of which can be used to distinguish the senders in the multipoint VC. This can make switching fast and connection management simple. However, AAL3/4 suffers from excessive overhead and is not well supported.

2. **VC mesh:** Another solution is to overlay one-to-many VCs to create many-to-many multicast, forming a VC mesh [1]. In this case, cells from different senders can be differentiated based on their VPI/VCI fields. This solution does not scale and requires N one-to-many VCs for N senders.

3. **Multicast servers (MCSs):** In this case, all senders send to the MCS which forwards data on a point-to-multipoint VC [15]. This approach is simple. The problem with it is it is inefficient, and the MCS needs large amounts of buffering. In addition, the MCS can become overloaded, which makes it difficult to guarantee quality of service requirements.

4. **Tokens:** Tokens can be used to coordinate senders. This approach is used in the SMART scheme [6, 7]. In this case, a sender must acquire a control message (token) before it can transmit data, and there is only one token in each VC. Hence, only one sender can transmit at a time, and no cell interleaving can possibly occur. Although this mechanism is feasible, the overhead and delay of the scheme are high.

5. **VC merge:** The VC merge approach entails packet-level buffering. This is achieved by buffering cells of other packets at the switch until all cells of the current packet go through (as shown in figure 2). The technique is also called "cut-through forwarding," and it is used

in the SEAM [8] and ARIS [19] schemes. It entails the implementation of a packet-based scheduling algorithm at the merging point, and maintaining separate queues for each sender. The AAL5 end-of-message bit is used to signal to the switch that a packet from a different port can now be forwarded. This approach is extremely fast and simple. However, it may require more memory at the switches, and may add to the burstiness and latency of traffic.

6. **VP merge:** This approach uses multipoint Virtual Paths (VPs). Here, only the VPI field is used for switching cells of a multipoint connection, and the VCI field is used to uniquely identify the sender. Connection management is simple in this case, but the approach requires receivers to have static assignment of VCs within VPs. In addition, VPs are preferred not be used by end-systems, since network providers use VPs for aggregation in the backbone. Finally, there are only $2^{12} = 4096$ unique VPI values possible at each hop.

7. **Variable VP merge:** Here, different VPI field sizes are used. [16, 17, 18]. The switches support both 12-bit VPI fields, as well as 18-bit VPI fields. Distributed schemes to assign globally unique VCIs within each VP are proposed using collision avoidance in [16]. This approach overcomes the VP scarcity problem of VP merge, but still has the problem of using VPs, and it also complicates the switch design since two VP tables need to be maintained.

8. **Sub-channel multiplexing:** A sub-channel is a "channel within a VC." Each sub-channel can be assigned an identifier called the sub-channel number to distinguish between multiple sub-channels in a VC [2]. Four bits from the Generic Flow Control (GFC) bits in the ATM cell header can carry this number. Each burst of cells is preceded by a "start" control (RM) cell and followed by an "end" RM cell. The sub-channel is allocated on the "start" cell and released on the "end" cell. Sub-channel identifiers can change at every switch. This approach allows dynamic sharing by using on-the-fly mapping of packets to sub-channels. However, four bits allow only up to fifteen concurrent senders (hexadecimal FF indicates an idle sub-channel). If no sub-channel is available, the burst of cells is lost, so this solution may not be scalable.
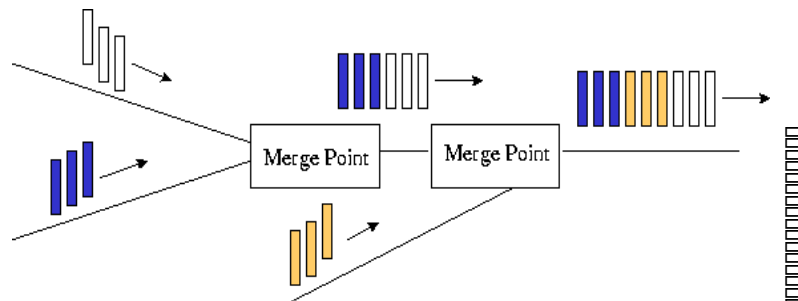

Figure 2: The VC merge approach

This paper emphasizes the issues involved if VC merge and VP merge are implemented.

3   Related Work

Little work has been done to define traffic management rules for multipoint-to-point connections [9]. Multipoint-to-point connections require feedback to be returned to the appropriate sources at the appropriate times. As illustrated in figure 3, the bandwidth requirements for a VC after a merge point is the sum of the bandwidths used by all senders whose traffic is merged. This is because the aggregate data rate after a merging point is the sum of all incoming data rates to the merging point [11]. Similarly, the number of RM cells after merging is the sum of those from different branches. Hence, the ratio of RM to data cells remains the same.
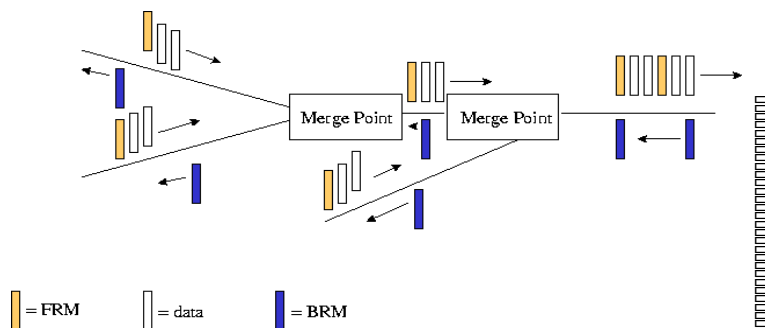

Figure 3: Multipoint-to-point connections

Ren and Siu [12, 13] describe an algorithm for multipoint-to-point congestion control, which allows senders belonging to the same connection to send at different data rates. The algorithm assumes that a multipoint-to-point VC is defined as a shared tree, and that VC merging is employed to prevent the cell interleaving problem. The authors proved that if the original point-to-point switch algorithm is max-min fair, the multipoint-to-point version is also max-min fair *among sources* [12, 13] (and not VCs).

The idea of Ren and Siu's algorithm is very similar to point-to-multipoint algorithms (see [4]). When a forward resource management (FRM) cell originating at a leaf is received at the merge point, it is processed and forwarded to the root. The merge point also returns a backward resource management (BRM) cell to the source which sent the FRM cell. The explicit rate in the BRM cell is set to the value of a register called MER (minimum explicit rate), maintained at the merge point. The MER register is then reset to the peak cell rate. The receipt of a BRM cell at the merge point simply triggers the normal processing of the BRM cell, and the ER value the BRM contains is used to set the MER register (the register is set to the minimum of its current value and the value of the ER in the BRM cell). The BRM cell is then discarded.

Another alternative is to maintain a bit at the merge point for each of the flows being merged [14]. The bit indicates that an FRM has been received from this flow after a BRM was sent to it. Therefore, when an FRM is received at the merge point, it is forwarded to the root and the bit is set, but the RM cell is not turned around as before. 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. This saves the overhead that the merge point incurs when it turns around RM cells, since only destinations turn around RM cells in this case [14].

4   Fairness for Multipoint-to-Point Connections

In this section, we define different types of fairness, and show an example of their operation. In addition, we discuss the merits and drawbacks of each type.

4.1   Possible Fairness Definitions

If a single N-to-one connection is treated as N one-to-one connections (VCs), the max-min fairness definition can be easily extended to achieve fairness among **sources**, regardless of which VC each source belongs to. We call this source-based fairness. Note that if multipoint VCs employ the same VPI/VCI for each multipoint conversation on a certain hop, and implement VC merge at the switches, there is no way for a switch to determine the number of sources in the same multipoint VC, or maintain any type of per-source accounting information.

Observe, however, that with source-based fairness, VCs that have a larger number of concurrently active senders get more bandwidth than VCs with less concurrent senders on the same link. Thus the resource allocation is not max-min fair **among the VCs**. If VC-based max-min fairness is required, then bandwidth allocation must be max-min among VCs, and allocations to the sources in the same VC can be max-min fair within the VC. This can be done in several ways that will be explained.

A third possibility is **flow-based** max-min fairness. Intuitively, each VC coming on an input port (link) is considered a separate flow. Hence, two VCs coming on the same input port are considered two separate flows, and traffic coming from two different input ports on the same VC (and being merged at the switch) is also considered as two separate flows. *They key point is that a switch can easily distinguish the flows.*

Formally, we define a *flow* for an output port as the sum of the number of VCs sending to this output port, for each of the input ports of the switch:

$NumFlows_j, j \in OutputPorts =$
$\quad \forall i, i \in InputPorts, \sum_i$ Number of VCs coming on port i and being switched to port j

For example, if traffic is coming from three different input ports (ports 1, 2, and 3) and is being switched to the same output port (port 4), and one of the input ports (port 2) has two VCs sending to port 4, while each of the other two ports (ports 1 and 3) has only one VC sending to port 4 (may

be the same VC, but different senders), then the number of flows at port 4 would be considered as 2 (port 2), plus 1 (port 1), plus 1 (port 3), equals four. The flow-based max-min fairness divides bandwidth fairly among the active flows. We will later see that this definition suffers from some drawbacks. The flow-based definition can be also be adopted within each VC in the VC-based approach, as seen in the next example.

The examples presented next will clarify the differences between the various fairness definitions.

4.2 Examples

We explain the different ways of defining fairness in multipoint situations with the aid of two examples. The first example illustrates a downstream bottleneck situation, while the second one shows an upstream bottleneck, to illustrate the allocation of capacity left-over by connections bottlenecked elsewhere.
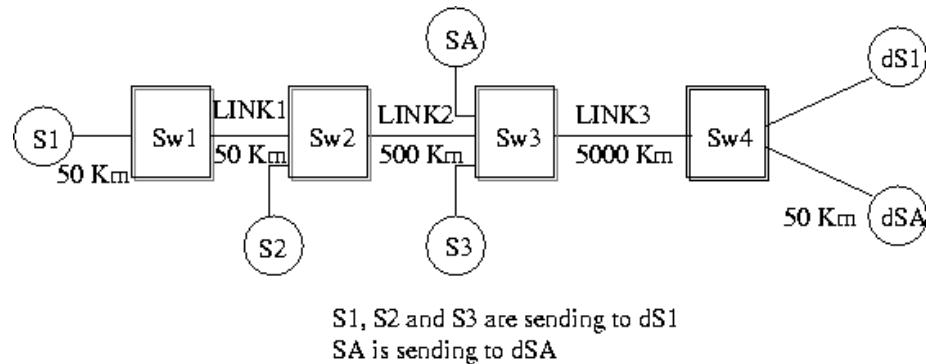
4.2.1 Example 1

Figure 4: Example multipoint-to-point configuration with a downstream bottleneck

Figure 4 illustrates a configuration with two VCs: one of the VCs is a multipoint-to-point VC with three senders and one receiver, 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 approximately 150 Mbps (after SONET overhead is accounted for). Clearly, all four sources are sharing a bottleneck link ($LINK_3$) between $Switch_3$ and $Switch_4$. **The aim of this example is to show the division of the 150 Mbps capacity of this bottleneck link among the sources.**

Source-based Definition.   In this case, we disregard which sources belong to which connections, and simply treat this as a normal "four sources on a single bottleneck" situation. Applying the max-min fairness definition among sources, the allocations computed are:

8

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

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

Observe, however, that on $LINK_3$, the multipoint-to-point VC is getting 3 times as much bandwidth as the point-to-point VC. If there were 100 concurrent senders in the multipoint-to-point VC, it would get 100 times as much bandwidth as the point-to-point VC. In essence, the bandwidth allocated to a multipoint-to-point VC with N concurrent senders 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-sender multipoint-to-point VC bottlenecked on the same link.

VC-based Definition: VC/Source. If a VC-based definition is adopted, we are essentially dividing up the max-min fair allocation computation process into two phases. In the first phase, we ignore the number of senders in each VC, and simply count the VCs bottlenecked at each node, applying the max-min fairness computation. In the second phase, we take each multipoint-to-point VC separately and divide up its allocation max-min fairly among the senders in that VC. This process is repeated for each multipoint-to-point VC.

According to this definition, the allocation vector for the example above would be:

$\{S_1, S_2, S_3, S_A\} \leftarrow \{25, 25, 25, 75\}$

This is because both of the VCs are bottlenecked at $LINK_3$, so each VC is allocated half of the available bandwidth ($\frac{1}{2} \times 150 = 75$). Then, for the multipoint-to-point VC, we see that $LINK_3$ is again the bottleneck, so each of the three sources gets one third of the bandwidth allocated to this VC ($\frac{1}{3} \times 75 = 25$).

Flow-based Definition. Recall that a flow was defined as a VC coming on an input port. According to this definition, the number of flows on $LINK_3$ is three, and hence each of the flows gets one third of the bottleneck bandwidth ($\frac{1}{3} \times 150$). This bandwidth is then divided equally among the two flows seen at the output port of $Switch_2$, producing the allocation vector:

$\{S_1, S_2, S_3, S_A\} \leftarrow \{25, 25, 50, 50\}$

Clearly, this suffers from the "beat-down problem" commonly observed in EFCI situations. This means that sources whose flow travels a larger number of hops are allocated less bandwidth than those travelling a smaller number of hops, even if both flows have the same bottleneck. In flow-based fairness, sources whose flow crosses a larger number of merge points are allocated less bandwidth than those crossing a smaller number of merge points.

VC-based Definition: VC/Flow. If we use the VC-based approach, but, instead of dividing the bandwidth among the senders in the same VC max-min fairly, we divide the bandwidth max-

min fairly among the flows in the VC, a different allocation vector is obtained. For the example above, the allocation vector would be:

$$\{S_1, S_2, S_3, S_A\} \leftarrow \{18.75, 18.75, 37.5, 75\}$$

This is because the bandwidth is divided max-min fairly among the two VCs at $Switch_3$, giving 75 Mbps to each VC. For the multipoint-to-point VC, $Switch_3$ divides the 75 Mbps equally among the two flows in that VC, so the flow originating from S_3 is allocated $\frac{1}{2} \times 75 = 37.5$ Mbps. $Switch_2$ divides the 37.5 Mbps that $Switch_3$ had allocated to the flow consisting of S_1 and S_2 equally among these two sources, each getting $\frac{1}{2} \times 37.5 = 18.75$ Mbps.

4.2.2 Example 2

S1, S2, S3 and S4 are sending to dS1
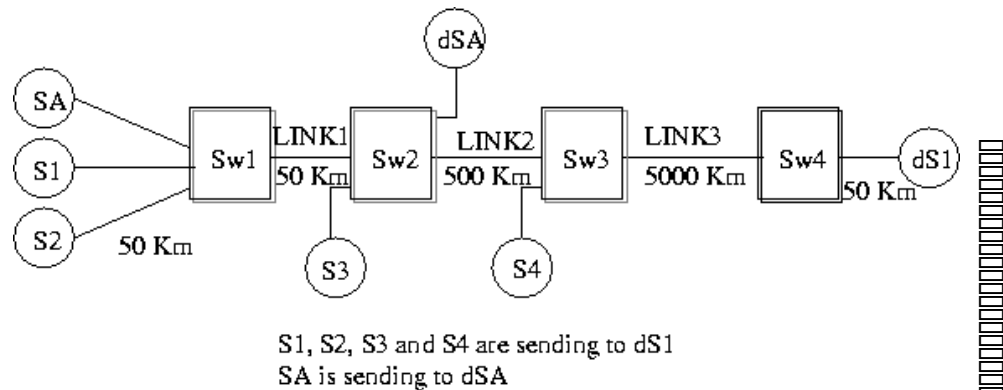SA is sending to dSA

Figure 5: Example multipoint-to-point configuration with an upstream bottleneck

Figure 5 illustrates a configuration with two VCs: one of the VCs is a multipoint-to-point VC with four senders and one receiver, 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 $LINK_3$.**

Source-based Definition. 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 upon sources S_3 and S_4.

VC-based Definition: VC/Source. According to the VC/Source definition, the allocation vector for the example above would be:

$$\{S_1, S_2, S_3, S_4, S_A\} \leftarrow \{12.5, \ 12.5, \ 62.5, \ 62.5, \ 25\}$$

This is because each of the VCs is allocated half of the bandwidth on $LINK_1$, and this bandwidth is divided equally among S_1 and S_2 of the multipoint VC. On $LINK_3$, the remaining capacity $(150 - 25 = 125$ Mbps) is divided max-min fairly among the sources within the multipoint-to-point VC.

Flow-based Definition. Here the allocation vector is:

$$\{S_1, S_2, S_3, S_4, S_A\} \leftarrow \{16.67, \ 16.67, \ 41.67, \ 75, \ 16.67\}$$

This is because $Switch_3$ sees two flows on $LINK_3$, and allocates half of the capacity to each flow (hence, source S_4 is allocated half of $LINK_3$ bandwidth). $Switch_1$ divides the 50 Mbps equally among the three flows sharing $LINK_1$ (each of S_1, S_2 and S_A gets $\frac{1}{3} \times 50 = 16.67$). $Switch_2$ divides the 75 Mbps (that $Switch_3$ had allocated to the flow emerging from it) equally among the flow from S_3 and the flow from $Switch_1$, but detects that one of the flows (that from $Switch_1$, i.e., S_1 and S_2) is only using 33.33 Mbps, so it allocates the remaining $75 - 33.33 = 41.67$ Mbps to source S_3.

VC-based Definition: VC/Flow. According to the definition, the allocation vector for this case is:

$$\{S_1, S_2, S_3, S_4, S_A\} \leftarrow \{12.5, \ 12.5, \ 50, \ 75, \ 25\}$$

$Switch_1$ divides the available 50 Mbps equally among the two VCs (giving each 25 Mbps), and divides the bandwidth of the multipoint-to-point VC equally among the two flows in that VC (each getting 12.5 Mbps). $Switch_3$ divides the bandwidth fairly among the two flows, allocating 75 Mbps to the flow from S_4 and 75 Mbps to the flow from $Switch_2$. $Switch_2$ sees that the flow from S_1 and S_2 is only using 25 Mbps, so it allocates the remaining 50 Mbps to the other flow (source S_3).

4.3   Merits and Drawbacks of the Different Definitions

The two examples above show how fairness based upon the concepts of source, VC, and flow give quite different allocations in some situations.

First let us consider source-based fairness versus VC/source-based fairness. The source-based fairness completely ignores the membership of different sources to connections, and divides the available bandwidth max-min fairly among the sources currently active. If billing and pricing is based upon

sources, it can be argued that this mechanism a good fairness method, since allocation is fair among sources.

However, if pricing is based on connections (VCs), it does not make sense for a VC with 100 concurrent senders to be allocated 100 times the bandwidth of a point-to-point connection bottlenecked on the same link. Source-based fairness is clearly unfair if this is the billing method adopted, and VC/source-based fairness is better.

The flow-based method is not max-min fair if we view an N-to-one connection as N one-to-one connections, since the same flow can combine more than one source. However, we can argue that it may be better to favor sources crossing a smaller number of merge points, since these are more likely to encounter less bottlenecks anyway. For example, if a user in New York City is fetching some web pages from a server in Germany, he expects to wait longer than if he is fetching pages from within New York City.

Thus, although flow-based fairness may be unfair to sources whose traffic is merged many times with other flows, this might be acceptable in many practical situations. The VC/flow-based fairness is max-min fair with respect to VCs, but within the same VC, it favors sources whose traffic goes through a smaller number of merge points.

The next two sections discuss the complexity of the design and implementation of algorithms to compute the above mentioned allocations.

5   Common Multipoint Algorithm Design Issues

There can be several different ways to implement multipoint-to-point ABR flow control algorithms. Each method offers a tradeoff in fairness, complexity, scalability, overhead and response time. Some of these issues are summarized next. Section 6 further discusses the design of multipoint algorithms.

- **VC merge versus VP merge.** With *VC merge* implementations (see section 2), it is impossible to distinguish among the cells of different sources in the same multipoint-to-point VC (since the same VPI/VCI fields are used for all the cells of a VC on the same hop). Hence, switch traffic management algorithms must not rely on being able to determine the number or rates of *active sources* with VC merge (number and rates of active VCs and number and rates of active flows can still be determined). With *VP merge*, however, the VCI field is used to distinguish among cells of different sources in the same multipoint-to-point VC on the same hop. Hence, it is possible to determine the number and rates of *active sources* in such implementations, and perform any necessary per-source accounting operations. (This, however, may incur additional complexity and reduce scalability.)

- **Per-source/VC/flow accounting.** All switch traffic management algorithms need to use some registers for storing the values they need to compute the rate allocations. Some of these values are stored for each input port, and some for each output port. Other algorithms use per-VC accounting, per-source accounting, or per-flow accounting. With multipoint-to-point VCs, per-VC accounting, per-source accounting, and per-flow accounting are no longer equivalent (they are equivalent for point-to-point scenarios). This leads to a set of interesting problems. For example, some algorithms store the value of the current cell rate (CCR) indicated in FRM cells, and later use it for computation. But the CCR value should be stored per-source, and the sources cannot be distinguished with VC merge. Other algorithms also attempt to measure the source rate of senders, or distinguish between overloading and underloading sources (e.g., MIT scheme, UCSC, and ERICA schemes). This is also infeasible with VC merge. *In general, per-source accounting is infeasible with VC merge, while per-VC accounting must account for the VC as a whole (even if its traffic is coming from different ports), and per-flow accounting must distinguish both input ports and VCs.*

- **Using downstream rate allocations.** For point-to-point connections, and multipoint-to-point connections when using source-based fairness, the switch computes the rate allocations it can support, and then indicates these allocations in the BRM cells only if they are less than the allocations computed by downstream switches (as indicated in the ER field of BRM cells). This suffices for these situations since the algorithm operates at the source level only, and all sources at a bottleneck are allocated equal rates. With VC/source, flow, and VC/flow-based fairness, however, downstream switches compute aggregate rate allocations that must be further subdivided among senders in upstream switches. Thus, the switches must use the downstream rate allocations as an estimate of the maximum available capacity for the VC/flow.

- **Which component generates the BRM cells** (i.e. turns around the FRM cells)? Should the merge point, or should the destination perform this operation? If the merge point turns around the BRM cells, the scheme may incur more overhead.

- Some merge point algorithms **wait for an FRM cell** to be received before sending feedback. What are the implications of this on the scalability of the scheme? Will the feedback delay grow with the number of levels of merge points? If you have to wait for the next FRM cell at each of the merge points, the time to return a BRM cell can increase with the number of levels of the tree, which is an undesirable property. This is also dependent on the FRM cell rate, the BRM cell rate, and their relationships during transient phases. Schemes that return the BRM cell received from the root, to the leaves which have sent FRM cells to the merge point since the last BRM cell was passed, are less sensitive to number of merge points.

6 Implementation of Algorithms for each Approach

In section 4, we discussed four different types of fairness that can be defined for multipoint VCs. This section discusses how switch traffic management algorithms need to be adapted to compute the fair allocations for each type.

1. **Source-based fairness.**

   This type of fairness is the easiest to design and implement, since it is an extension of point-to-point algorithms. The algorithm gives the same allocation to all sources bottlenecked on the same link, and it only operates at the source level. However, source-based fairness in VC merge implementations poses some problems, since sources in the same VC cannot be distinguished. The main considerations for switch algorithms in this case is to avoid any per-source accounting and any attempt to estimate the number or rates of active sources. Note, however, that such changes may result in some oscillations and slow transient response for most algorithms, since per-source accounting and estimation of the number and rates of active sources can considerably improve switch algorithm performance.

2. **VC/Source-based fairness.**

   VC/source-based fairness is not a straightforward extension of point-to-point algorithms, since the algorithm has to operate at two different levels: the VC level and the source level. Fair allocation of bandwidth to VCs can be simple, since VCs can be easily distinguished, their rates estimated, and the ABR available capacity can be easily measured. As with source-based fairness, VC merge implementations imply that allocations must not depend on any source-level metrics. Additional complexity is introduced by the two-level operation, which necessitates estimation of the load and capacity at both the link level and the VC level. Hence, it becomes necessary to use the explicit rates assigned by downstream switches for the VC in making allocations at upstream switches.

3. **Flow-based fairness.**

   Flow-based fairness is also non-trivial to implement. The capacity needs to be fairly divided upon the currently active flows at every node. This introduces the need for bilevel computations, since *two separate flows can be merged into one flow at any node*. If needed, counting the number of flows for each output link is a straightforward task, since a single bit can be maintained for each VC on each input port. If a counter maintains the number of bits set for connections to be switched to each output port, this number is the same as the number of flows on the output link. Hence, the number of active flows can be measured over successive intervals, and exponential averaging can be used to smooth out the value. Alternatively, the activity level of each flow can be estimated as the ratio of the rate of this flow and the maximum share a flow can get. The main concern for flow-based fairness, however, is that

the bottleneck capacity available for a flow needs to be carefully estimated, since it depends on the explicit rate value that downstream switches allocate to the flows emerging from the switch being considered.

4. **VC/Flow-based fairness.**

   As with VC/source-based fairness, VC/flow-based fairness must operate at two different levels: the VC level and the flow-within-a-VC level. Distinguishing among VCs, and among different flows within the same VC are both quite simple. However, computing the actual allocations in a distributed manner may not be extremely straightforward, since information from downstream switches is needed, and handling the two-level operation introduces additional complexity.

7 Summary and Conclusions

There are several issues to be resolved in ATM multipoint communication, including devising a scalable method for merging traffic from multiple senders, and several traffic management problems.

Four different types of fairness can be defined for multipoint-to-point connections:

1. **Source-based fairness,** which divides bandwidth fairly among active sources as if they were sources in point-to-point connections, ignoring group memberships.

2. **VC/source-based fairness,** which first gives max-min fair bandwidth allocations at the VC level, and then fairly allocates the bandwidth of each VC among the active sources in this VC.

3. **Flow-based fairness,** which gives max-min fair allocations for each active flow, where a flow is a VC coming on an input link. Formally,
   $NumFlows_j, j \in OutputPorts =$
   $\forall i, i \in InputPorts, \sum_i$ Number of VCs coming on port i and being switched to port j

4. **VC/flow-based fairness,** which first divides the available bandwidth fairly among the active VCs, and then divides the VC bandwidth fairly among the active flows in the VC.

Design issues common to multipoint traffic management algorithms include minimizing overhead and delays, use of VP merge versus VC merge, use of downstream allocations, and, most importantly, the use of per-source accounting, per-VC accounting and per-flow accounting in switch algorithms. Since sources, VCs, and flows are equivalent for point-to-point connections, but different for multipoint-to-point connections, it is important to note the differences between the three types of accounting. Per-source accounting cannot be performed in VC merge implementations,

and can only be performed with VP merge. Per-flow accounting has to distinguish VCs and input ports, while per-VC accounting must combine the VC information coming from different input ports.

Modifications are necessary for switch algorithms to implement each of the four types of fairness. For source-based fairness (the simplest), algorithms operating with VC merge should not attempt any source-level accounting, and must only use information supplied in the RM cells, in addition to aggregate measurements of load, capacity and queuing delays. VC/source-based fairness must make VC-level allocations and source-level allocations, making use of per-VC accounting. Flow-based fairness can be achieved by estimating flow activity and available flow capacity, and VC/flow-based fairness should also estimate both VC and flow load and capacity.

It is essential to continue this work to define the desirable forms of fairness, and extend current switch traffic management algorithms for multipoint connections. Extensive performance analysis is also crucial to examine the fairness, complexity, overhead, transient response, delays, and scalability tradeoffs involved.

References

[1] G. Armitage. Support for multicast over UNI 3.0/3.1 based ATM networks. Request for Comments 2022, November 1996.

[2] Milind M. Buddhikot and Christos Papadopoulos. Washington university workshop on integration of IP and ATM: Overview of session 3: Multicast support in ATM+IP networks. http://www.arl.wustl.edu/arl/workshops/atmip/proceedings.html, 1996.

[3] C. Diot, W. Dabbous, and J. Crowcroft. Multipoint communication: A survey of protocols, functions and mechanisms. *IEEE Journal on Selected Areas in Communications*, 15(3):277–290, April 1997.

[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. ATM Forum/97-0615, July 1997.

[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] Eric Gauthier, Jean-Yves Le Boudec, and D. Dykeman. SMART: A many-to-many multicast protocol for ATM. ATM Forum/96-1295, October 1996.

[7] Eric Gauthier, Jean-Yves Le Boudec, and Philippe Oechslin. Shared many-to-many ATM reservations. In *Proceedings of IEEE ATM'96 Workshop, San Francisco*, August 1996.

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

[9] Juha Heinanen. Multipoint-to-point VCs. ATM Forum/97-0261, April 1997.

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

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

[12] Wenge Ren, Kai-Yeung Siu, and Hiroshi Suzuki. Performance evaluation of multipoint-point ABR and UBR. ATM Forum/96-1402, October 1996.

[13] 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.

[14] Wenge Ren, Kai-Yeung Siu, Hiroshi Suzuki, and Masayuki Shinihara. Multipoint-to-multipoint ABR service in ATM. *Submitted to IEEE Journal on Selected Areas in Communications*, 1997.

[15] Rajesh R. Talpade, Grenville J. Armitage, and Mostafa H. Ammar. Experience with architectures for supporting IP multicast over ATM. In *Proceedings of IEEE ATM'96 Workshop, San Francisco*, August 1996.

[16] R. Venkateswaran, C. S. Raghavendra, Xiaoqiang Chen, and Vijay P. Kumar. Distributed mechanisms for VCI assignment in VP-based multicast. ATM Forum/97-0379, April 1997.

[17] R. Venkateswaran, C. S. Raghavendra, Xiaoqiang Chen, and Vijay P. Kumar. Support for group multicast in PNNI. ATM Forum/97-0076, February 1997.

[18] R. Venkateswaran, C. S. Raghavendra, Xiaoqiang Chen, and Vijay P. Kumar. Support for multiway communications in ATM networks. ATM Forum/97-0316, April 1997.

[19] A. Viswanathan, N. Feldman, Rick Boivie, and Rich Woundy. ARIS: Aggregate route-based IP switching. IETF Internet Draft: draft-viswanathan-aris-overview-00.txt, March 1997.

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