CHAPTER 3
FEEDBACK CONSOLIDATION ALGORITHMS FOR POINT-TO-MULTIPOINT CONNECTIONS

The traffic management problem for point-to-multipoint connections is an extension to the traffic management for unicast connections. However, some additional problems arise in the point-to-multipoint case. In particular, the consolidation of the feedbcak information from the different leaves of the tree is necessary for point-to-multipoint connections. This is because of the "feedback implosion" problem (feedback information provided to the sender should not increase proportional to the number of leaves in the connection). Scalability becomes a major concern.

Many general frameworks have been suggested that convert any unicast congestion control switch algorithm to work for point-to-multipoint connections. This chapter highlights the major issues and ideas in this area.

3.1 INTRODUCTION

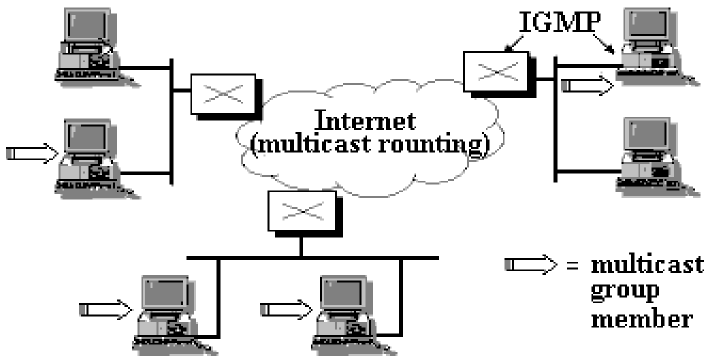

Figure 3.1: Multicasting in the Internet

Multipoint communication is the exchange of information among multiple parties. Examples of multipoint applications include audio and video conferencing applications, video on demand, distance learning, distributed games, server and replicated database synchronization, advertising, searching and data distribution applications. Multipoint capabilities are essential for ATM networks to efficiently support many applications, including IP multicasting and overlay applications (see figure 3.1). The ABR service category attempts to provide possibly non-zero minimum rate guarantees, achieve fairness, and minimize cell loss for data (non real-time) traffic by periodically indicating to ABR sources the rate at which they should be transmitting.

ABR traffic control (discussed in the previous chapter) requires the sources to send at the rate specified by the network in feedback (resource management) cells. For point-to-multipoint connections, feedback consolidation at the branch points becomes necessary. The operation of feedback consolidation can be explained by figure 3.2.

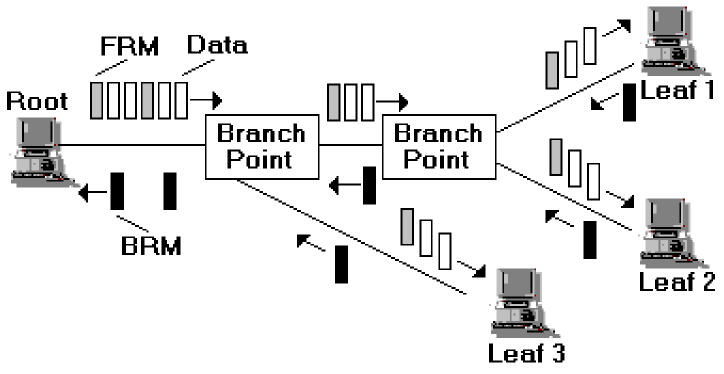

Figure 3.2: ABR Point-to-Multipoint Connections

The consolidation operation avoids ***the feedback implosion problem***, where the number of backward resource management (BRM) cells received at the source is proportional to the number of leaves in the multicast tree. In addition, the allowed rate of the source should not fluctuate due to the varying feedback received from different leaves.

In point-to-point ABR connections, the source transmits at the minimum rate that can be supported by all the switches on the path from the source to the destination [4]. The natural extension of this strategy for point-to-multipoint connections is controlling the source to the minimum rate that can be supported by the switches on the paths from the source to all of the leaves in the multicast tree, as shown in figure 3.3.

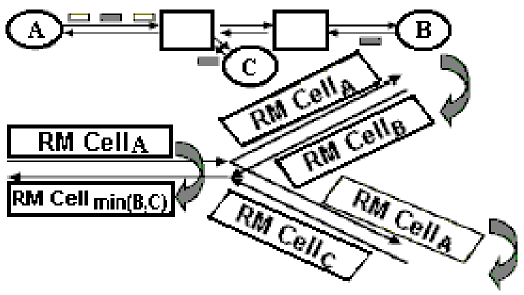

Figure 3.3: Taking the minimum in point-to-multipoint ABR connections

The minimum rate is the technique most compatible with typical data requirements: no data should be lost, and the network can take whatever time needed for data delivery.

A serious problem may arise if the BRM sent by the branch point towards the root doesn't consolidate feedback information from all branches [12]. This will introduce noise called "***consolidation noise***" leading to rate oscillation at the root.

A number of consolidation algorithms has been proposed in [13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. Several design and implementation considerations come into play when developing a consolidation algorithm. The oscillations and transient response of the algorithm are important. The algorithm should preserve the efficiency and fairness properties of the rate allocation schemes employed in the network switches. It must also be scalable to very large multicast trees. The implementation complexity, feedback delay, and the overhead of the BRM cells should not be increased with the increase of the number of levels or leaves of the multicast tree. Handling non-responsive branches is necessary such that they neither halt the consolidation operation nor cause overload (or underload).

3.2 DESIGN ISSUES

There are several ways to implement the feedback consolidation algorithm at branch points. Each method offers a tradeoff in complexity, scalability, overhead, transient response, and consolidation noise. The tradeoffs can be summarized as follows:

[A] Which component generates the BRM cells (i.e., turns around the FRM cells)? Should the branch point, or should the destination, perform this operation?

[B] What is the condition to trigger sending a BRM at the branch point? Should the branch point wait for feedback from all the branches before passing the BRM cell upstream? Although this eliminates the consolidation noise, it incurs additional complexity, and increases the transient response of the scheme (especially after idle or low rate periods).

[C] Does the scheme scale well? How can the ratio of FRM cells generated by the source to BRM cells returned to the source be controlled? Will the feedback delay grow with the number of branches? For example, if the algorithm waits for an FRM cell to be received before sending feedback, the delay might increase with the number of levels of the multicast tree.

[D] How does the branch point operate when it is also a switch (queuing point)? The coupling of the switch and branch point functions must be considered. When should the actual rate computation algorithm be invoked?

[E] How are non-responsive branches handled? If the consolidation scheme waits for feedback from all the branches before sending a BRM to the source, an algorithm must be developed to determine when a branch becomes non-responsive and handle this case. A timeout or outstanding RM cell count mechanism must be implemented.

[F] How is accounting performed at the branch point? Consolidation algorithms use registers to store values such as the minimum rate given by branches in the current iteration, and flags to indicate whether an RM cell has been received since the last one was sent. Some values, if stored per output port, need an efficient data structure. For example, if the rate returned by each branch is stored (instead of only maintaining the minimum value for this connection for this round), a heap is necessary to enable the minimum operation to be rapidly performed.

3.3 CONSOLIDATION ALGORITHMS

This section describes some previously proposed consolidation algorithms. In the algorithms presented, a point-to-point rate allocation switch algorithm (like ERICA as explained in the previous chapter) is employed immediately before sending a BRM on the link. This ensures that the most recent feedback information is sent. Note that we omit handling of CI and NI flags in all algorithms for simplicity. The algorithms at the branch point operate as explained in the following subsections.

3.3.1 Robert's Algorithm [1994]

The main idea of the algorithm [13] is that BRM cells are returned from the branch point when FRM cells are received, and the BRM cells contain the minimum of the values indicated by the BRM cells received from the branches after the last BRM cell was sent. FRM cells are duplicated and multicast to all branches at the branch point.

A register, MER is maintained for each multipoint VC. MER stores the minimum of the explicit rate (ER) values indicated in the BRM cells which were received after the last BRM cell was sent. MER is initialized to the peak cell rate. A temporary variable MXER is also used when a FRM cell is received (its value doesn’t persist across invocations of the algorithm). It stores the ER from the FRM cell. The algorithm operates as follows.

```
Upon the receipt of an FRM cell:
1. Multicast FRM cell to all participating branches
2. Let MXER = ER from FRM cell
3. Return a BRM with ER = MER to the source
4. Let MER = MXER

Upon the receipt of a BRM cell:
1. Let MER = min (MER, ER from BRM cell)
2. Discard the BRM cell

When a BRM is to be scheduled:
Let ER = min(ER,ER calculated by rate allocation algorithm for all
branches)
```

This algorithm suffers from the consolidation noise problem when a BRM generated by a branch point does not consolidate feedback from all tree branches. In fact, if a BRM generated by the branch point does not accumulate feedback from any branch, the feedback can be given as the peak cell rate (if that branch point itself is not overloaded).

The method is also complex to implement because the switch has to turn around the RM cells. This is somewhat similar to the Virtual Source/Virtual Destination (VS/VD) concept [4]. Most studies argue that turning around RM cells has a high implementation complexity [18].

Although this algorithm may experience heavy consolidation noise due to inaccurate feedback and may lead to serious rate oscillation [15], it exhibits a very fast transient response. It also guarantees that the BRM to FRM ratio remains one since it generates a BRM for each received FRM.

3.3.2 Tzeng et al. Algorithm [1995]

Tzeng and Siu [14] have presented a slightly more conservative algorithm that remedies some of the inaccuracy problems of Robert's scheme. In the Tzeng and Siu algorithm, a switch will return a BRM cell to its upstream node only when it receives an FRM cell and it has received at least one BRM cell from a downstream node since the last time a BRM cell was delivered upstream at this switch. For this purpose, a boolean flag, AtLeastOneBRM (initially zero), is set to true when a BRM cell is received from a branch, and reset when a BRM is sent by the branch point. AtLeastOneBRM flag is stored for each multipoint VC. The explicit rate in the BRM cell is the same as in Robert's algorithm, i.e., the minimum of the explicit rate in the last FRM cell and all explicit rates reported to the switch since the last BRM was generated. The algorithm operates as follows.

```
Upon the receipt of an FRM cell:
1. Multicast FRM cell to all participating branches
2. IF (AtLeastOneBRM) THEN
       A. Let MXER = ER from FRM cell
       B. Return a BRM with ER = MER to the source
       C. Let MER = MXER
       D. Let AtLeastOneBRM = 0

Upon the receipt of a BRM cell:
1. Let AtLeastOneBRM = 1
```

```
2. Let MER = min (MER, ER from BRM cell)
3. Discard the BRM cell

When a BRM is to be scheduled:
Let ER = min(ER,ER calculated by rate allocation algorithm for all
branches)
```

Using this algorithm, more accurate feedback (than the previous algorithm) can be achieved and consolidation noise can be reduced somewhat. It also maintains the BRM to FRM ratio to be approximately equal to one (It is strictly less than one since the first FRM will never be turned).

However, the response time is slower than in Robert's scheme, since the switch must wait for a BRM cell to arrive from downstream before reporting on local overload. This method is also complex to implement for the same reason of Robert's.

3.3.3 Ren et al. Algorithm 3 [1996]

Ren et al. [15] have argued that the algorithms described so far may introduce heavy RM cell processing overhead on switches. Therefore, they propose an algorithm in which switches do not initiate generation of BRM cells, but instead simply forward selected BRM cells received from downstream, after modifying the contents. For example, Algorithm 3 in [15] will forward the first BRM cell received after an FRM cell has been received. The explicit rate reported in this cell will be the minimum of the peak cell rate (PCR) and all explicit rates reported to the switch since the last BRM cell was forwarded. A boolean flag, AtLeastOneFRM, indicates that a FRM cell has been received by the branch point after the last BRM cell was passed to the source. Again MER and AtLeastOneFRM are stored per multipoint VC. The algorithm operates as follows.

```
Upon the receipt of an FRM cell:
1. Multicast FRM cell to all participating branches
2. Let AtLeastOneFRM = 1

Upon the receipt of a BRM cell:
1. Let MER = min (MER, ER from BRM cell)
2. IF (AtLeastOneFRM) THEN
       A. Pass the BRM with ER = MER to the source
       B. Let MER = PCR
```

```
      C. Let AtLeastOneFRM = 0
   ELSE Discard the BRM cell

When a BRM is to be scheduled:
Let ER = min(ER,ER calculated by rate allocation algorithm for all
branches)
```

The behavior of this algorithm is similar to Tzeng and Siu's algorithm with reduced implementation complexity, and the problem of consolidation noise still exists. The BRM to FRM ratio is maintained to be less than or equal to one.

3.3.4 Ren et al. Algorithm 4 [1996]

A natural extension to the previous Ren et al. algorithm is for a switch to return a BRM cell only after receiving feedback information from all its downstream branches since the last BRM cell was sent (algorithm 4 in [15]). The content of this BRM cell is the minimum explicit rate reported in the BRM cells received from the downstream branches since the last BRM cell was sent.

To count the number of branches from which BRM cells were received at the branch point (after the last BRM cell was passed by the branch point), a counter, NBRMsRecv, is incremented the first time a BRM cell is received from each branch (NBRMsRecv is initialized to zero). As before, the MER and NBRMsRecv registers are maintained per multipoint VC.

Another counter, NBranches, stores the number of branches of the point-to-multipoint VC at this branch point and initialized during connection setup. In addition, if leaf initiated join is allowed (as in UNI 4.0), NBranches must be updated every time a new branch is added to a branch point.

A flag, BRMReceived, is needed for each branch to indicate whether a BRM cell has been received from this particular branch, after the last BRM cell was passed. The flag is stored for each output port and not for each VC, as it is needed for each branch. Note that a mechanism must be implemented to ensure that BRM cell flow is not stopped in the case of non-responsive branches. Timeouts or RM cell counters can be used for that purpose. The algorithm operates as follows.

```
Upon the receipt of an FRM cell:
Multicast FRM cell to all participating branches

Upon the receipt of a BRM cell from branch i:
1. IF (NOT $BRMReceived_i$) THEN
       A. Let $BRMReceived_i$ = 1
       B. Let NBRMsRecv = NBRMsRecv + 1
2. Let MER = min (MER, ER from BRM cell)
3. IF (NBRMsRecv = Nbranches) THEN
       A. Pass the BRM with ER = MER to the source
       B. Let MER = PCR
       C. Let NBRMsRecv = 0
       D. Let BRMReceived = 0 FOR all branches
   ELSE Discard the BRM cell

When a BRM is to be scheduled:
Let ER = min(ER,ER calculated by rate allocation algorithm for all
branches)
```

This extension can easily eliminate the consolidation noise, but the resulting slow transient response may not be satisfactory. Even when excessive overload is detected, the algorithm has to wait for feedback from (possibly distant) leaves before indicating the overload information to the source. By that time, the source may have transmitted a large number of cells (which would be dropped due to buffer overflows), resulting in performance degradation. This situation is especially problematic when the source has been idle for some time, and then suddenly sends a burst, so there are no RM cells initially in the network.

3.3.5 Moh's Algorithm [1997]

W. M. Moh has proposed in [16] an algorithm that presents new ideas. First, it considers the overload feedback from any branch as hot news that should not be delayed for the source. Overload is detected here when the feedback to be indicated is *less than or equal to* the current minimum rate. Second, the local congestion state of the switch plays a role in making the decision of returning BRM cells in that algorithm. Finally, It presents a timer as a solution for handling non-responsive branches.

In this algorithm, BRM cells are returned when FRM cells are received and one of the following conditions is satisfied:

C1: At least a new BRM cell with a requested rate less than or equal to the current rate is received.

C2: At least a new BRM cell has been received, and a FRM cell arrival triggers the switch to compute an explicit rate that is less than or equal to the current rate.

C3: BRM cells have been received from all branches.

C4: At least a new BRM cell has been received, and a controlled timing parameter Δ time-units have elapsed since the last BRM cell has been sent.

C1 and C2 ensure that a BRM with non-increasing rate will be honored promptly, C3 prevents premature rate-increase, and finally C4 prevents deadlock or infinite wait due to (temporarily or permanently) broken links which might be quite frequent in wireless networks.

To implement these four conditions, the algorithm use a flag READY_i for each branch i traversing this switch. Another flag READY indicates whether a BRM cell should be returned (as a result of the first or the third condition) or not. In addition, a timer, TIMER, which expires when the last BRM cell has been sent no less than Δ (representing the time-out period) time-units, is used to avoid deadlock. The algorithm operates as follows.

```
Upon the receipt of an FRM cell:
1. Multicast FRM cell to all participating branches
2. Let MXR =ER;
3. $ER_{sw}$ = min ER calculated by rate allocation algorithm for all branches
4. IF((READY=1) OR ((OR($READY_i$ for all i)) AND (($ER_{sw} \leq$ MER) OR (TIMER$\leq$0))))
   THEN
       A. Let MER = min (MER, $ER_{sw}$)
       B. Return a BRM with ER = MER to the source
       C. Let READY = 0
       D. Let $READY_i$ = 0 for all branches
       E. Let TIMER =$\Delta$
   ELSE Let MER = min (MER, ERsw)
5. Let MER = MXR

Upon the receipt of a BRM cell from branch i:
1. IF (ER$\leq$ MER) THEN
       A. Let READY = 1
   ELSE
       A. Let $READY_i$ = 1
       B. Let READY = AND ($READY_i$ for all i)
```

```
2. Let MER = min (MER, ER)
3. Discard this BRM cell.

When a BRM is to be scheduled:
Let ER = min(ER,ER calculated by rate allocation algorithm for all
branches)
```

The algorithm exhibits a fast transient response and has the advantage of handling the non-responsive branches. On the other hand, it has a high implementation complexity because of turning around FRM cell. It also suffers from consolidation noise because any feedback, which is less than MER, (even if MER is equal to PCR as the case immediately after sending a BRM cell) is returned to the source while it may not be the most bottleneck.

3.3.6 Fahmy, Jain et al. Algorithms [1998]

The main idea behind the algorithms presented by [18,32] is that the slow transient response problem should be avoided when a severe overload situation has been detected. In this case, there is no need to wait for feedback from all the branches, and the overload should be immediately indicated to the source. In cases of underload indication from a branch, it is better to wait for feedback from all branches, as other branches may be overloaded. This is somewhat similar to the idea behind the backward explicit congestion notification (BECN) cells sent by the switches. Overload is detected here when the feedback to be indicated is *much less* than the last feedback returned by the branch point. The "much less" condition is tested using a multiplicative factor, Threshold. The Threshold value can range from zero to one.

A Threshold value of one means that overload is detected when the feedback to be given is less than the current ER (even when it is 99% of the last ER value given); a Threshold value of zero means that overload is not detected except when the new rate to be indicated is zero, which, in effect, disables the fast overload indication feature.

Note that when a BRM cell is returned due to overload detection before feedback has been received from all branches, the counter and the register values are not reset.

3.3.6.1 Fast Overload Indication Algorithm

In this algorithm, the LastER register maintains the last explicit rate value returned by the branch point. LastER is stored per multipoint VC and compared to the value of MER in step 5 below.

Two temporary variables: SendBRM and Reset are used. SendBRM is set only if a BRM cell is to be passed to the source by the branch point. Reset is false only if a BRM cell is being used to indicate overload conditions, and hence the register values should not be reset. The algorithm operates as follows.

```
Upon the receipt of an FRM cell:
Multicast FRM cell to all participating branches

Upon the receipt of a BRM cell from branch i:
1. Let SendBRM = 0
2. Let Reset = 1
3. IF (NOT BRMReceived$_i$) THEN
       A. Let BRMReceived$_i$ = 1
       B. Let NBRMsRecv = NBRMsRecv + 1
4. Let MER = min (MER, ER from BRM cell)
5. IF (MER $\leq$ (Threshold * LastER)) THEN   (* overload is detected *)
       A. IF (NBRMsRecv < Nbranches) THEN
              1. Let Reset = 0
       B. Let SendBRM = 1
   ELSE IF (NBRMsRecv = Nbranches) THEN
            A. Let SendBRM = 1
6. IF (SendBRM) THEN
       A. Pass the BRM with ER = MER to the source
       B. IF (Reset) THEN
              1. Let MER = PCR
              2. Let NBRMsRecv = 0
              3. Let BRMReceived = 0 FOR all branches
    ELSE Discard the BRM cell

When a BRM is to be scheduled:
1.Let ER=min(ER,ER calculated by rate allocation algorithm for all
branches)
2.Let LastER = ER
```

3.3.6.2 RM Ratio Control Algorithm

The previous algorithm may increase the BRM cell overhead, as the ratio of source-generated FRM cells to BRM cells received by the source can exceed one. To avoid this problem, Fahmy et al. introduced the register SkipIncrease which is maintained for each multipoint VC (and initialized to zero). SkipIncrease is used to control the RM cell ratio. SkipIncrease is incremented whenever a BRM cell is sent before feedback from all the branches has been received. When feedback from all leaves indicates underload, and the value of the SkipIncrease register is greater than zero, this particular feedback can be ignored and SkipIncrease decremented. Note that the value of the SkipIncrease counter will not increase to large values, as the rate allocation algorithm (such as ERICA) arrives at the optimal allocation within few iterations, and the explicit rates computed cannot continue decreasing indefinitely. Analysis and simulations [18] have shown that the counter never exceeds small values and quickly stabilizes at zero. A maximum value can also be enforced by the algorithm, which operates as follows.

Upon the receipt of an FRM cell:

```
Multicast FRM cell to all participating branches
```

Upon the receipt of a BRM cell from branch i:

```
1. Let SendBRM = 0
2. Let Reset = 1
3. IF (NOT BRMReceived_i) THEN
       A. Let BRMReceived_i = 1
       B. Let NBRMsRecv = NBRMsRecv + 1
4. Let MER = min (MER, ER from BRM cell)
5. IF (MER≥ LastER AND SkipIncrease > 0 AND NBRMsRecv = Nbranches) THEN
       A. Let SkipIncrease = SkipIncrease – 1
       B. Let NBRMsRecv = 0
       C. Let BRMReceived = 0 FOR all branches
   ELSE IF (MER < (Threshold * LastER)) THEN
       A. IF (NBRMsRecv < Nbranches) THEN
              1. Let SkipIncrease = SkipIncrease + 1
              2. Let Reset = 0
       B. Let SendBRM = 1
   ELSE IF (NBRMsRecv = Nbranches) THEN
A. Let SendBRM = 1
6. IF (SendBRM) THEN
       A. Pass the BRM with ER = MER to the source
       B. IF (Reset) THEN
              1. Let MER = PCR
              2. Let NBRMsRecv = 0
              3. Let BRMReceived = 0 FOR all branches
   ELSE Discard the BRM Cell
```

When a BRM is to be scheduled:

```
1.Let ER=min(ER,ER calculated by rate allocation algorithm for all
branches)
2.Let LastER = ER
```

3.3.6.3 Immediate Rate Computation Algorithm

The last two algorithms can offer rapid congestion relief when an overload is detected in a branch of the multicast tree. They do not, however, account for the potential overload situation at the branch point itself: if the branch point is a switch (queuing point), the rate allocation algorithm is only performed when the BRM cell is about to be scheduled on the link. In cases when the branch point is itself a switch and queuing point, the immediate rate calculation option invokes the rate calculation algorithm whenever a BRM is received, and not just when a BRM is being sent. Hence overload at the branch point can be detected and indicated according to the fast overload indication option as previously described. Doing this, however, may involve some additional complexity.

The algorithm presented below is the same as the previous one, except for the addition of the rate allocation algorithm invocation.

Upon the receipt of an FRM cell:

```
Multicast FRM cell to all participating branches
```

Upon the receipt of a BRM cell from branch i:

```
1. Let SendBRM = 0
2. Let Reset = 1
3. IF (NOT $BRMReceived_i$) THEN
       A. Let $BRMReceived_i$ = 1
       B. Let NBRMsRecv = NBRMsRecv + 1
4. Let MER = min (MER, ER from BRM cell)
5. Let MER = min (MER, minimum ER calculated by rate allocation algorithm
for all branches)
6. IF (MER $\geq$ LastER AND SkipIncrease > 0 AND NBRMsRecv = Nbranches) THEN
       A. Let SkipIncrease = SkipIncrease – 1
       B. Let NBRMsRecv = 0
       C. Let BRMReceived = 0 FOR all branches
   ELSE IF (MER < (Threshold * LastER)) THEN
```

```
        A. IF (NBRMsRecv < Nbranches) THEN
               1. Let SkipIncrease = SkipIncrease + 1
               2. Let Reset = 0
        B. Let SendBRM = 1
    ELSE IF (NBRMsRecv = Nbranches) THEN
        A. Let SendBRM = 1
7. IF (SendBRM) THEN
        A. Pass the BRM with ER = MER to the source
        B. IF (Reset) THEN
               1. Let MER = PCR
               2. Let NBRMsRecv = 0
               3. Let BRMReceived = 0 FOR all branches
```

When a BRM is to be scheduled:

```
1.Let ER=min(ER,ER calculated by rate allocation algorithm for all
branches)
2.Let LastER = ER
```

One of the disadvantages of Fahmy et al. algorithms is that they are all threshold-very-sensitive. If the threshold value is low, they exhibit the same slow transient response as the wait-for-all technique. They are also lacking a technique for handling non-responsive branches.

3.3.7 Ammar et al. Algorithm [1998]

In the design of consolidation algorithms there are typically two conflicting objectives:

- Speed: A switch should convey any new information about changed rates towards the source quickly.
- Accuracy: The switch should insure the accuracy of aggregated feedback information by making sure it has heard from all downstream paths before it forwards aggregated information towards the source.

Schemes that attempt to maximize accuracy of feedback information tend to be slow in providing feedback to the source when the conditions in the network change. This results in a long transient period until the source is able to adjust its rate to accurately meet prevailing network conditions. Accuracy can be traded for speed by having a switch generates feedback information before it has all the necessary information from downstream paths. This can cause consolidation noise, which can result in heavy oscillation of the cell rate used by the source. Most consolidation algorithms attempt to maximize one side of the speed-accuracy tradeoff.

Ammar et al. [19] presented a probabilistic consolidation algorithm that can be tuned using probability parameters to span the speed-accuracy spectrum. For each FRM cell, extra BRM cells *may be* sent upstream during the consolidation procedure (i.e., before feedback from all downstream paths is received) so that any detected change in conditions can be passed back to the source more quickly. The value of probability p (that an overload feedback is sent) can be adjusted to navigate the speed-accuracy tradeoff. When $p=0$ then the scheme becomes equivalent to the wait-for-all algorithm and the scheme achieves maximum accuracy. When $p=1$ then the scheme achieves maximum speed and suffers high consolidation noise.

At a switch, the following data structures are kept for each multicast virtual connection: Beside the usual register MER, M counters are needed to count the number of BRM cells received form each branch since last (not extra) BRM cell sent. M is the number of (immediate) downstream branches associated with this switch in multicast tree. Each counter is initialized to zero. A Decrement flag DecER is also used to indicate whether a more restrictive explicit rate has been received since the last BRM cell was sent upstream. An extra BRM cell may be generated only if DecER is true. DecER is initialized to false. The algorithm operates as follows.

```
Upon the receipt of an FRM cell:
Multicast FRM cell to all participating branches

Upon the receipt of a BRM cell from branch i:
1. Increment by one the counter corresponding to the branch i
2. IF (ER from the BRM Cell < MER) THEN
       A. Let DecER = True
3. IF (all counters > zero) THEN
       A. Let MER = min (MER, ER from BRM cell)
       B. Pass the BRM with ER = MER to the source
       C. Decrement all counters by one
       D. Let MER = PCR
       E. Let DecER = False
   ELSE IF(RandomValue<p AND DecER) THEN
       A. Let MER = min (MER, ER from BRM cell)
       B. Pass the BRM with ER = MER to the source
       C. Let DecER = False
   ELSE
```

```
       A. Let MER = min (MER, ER from BRM cell)
       B. Discard this RM cell

When a BRM is to be scheduled:
Let ER = min(ER,ER calculated by rate allocation algorithm for all
branches)
```

Ammar et al. proposed using a value of p = 1/M. Thus, on average one extra BRM cell will be sent during consolidation, regardless of the number of downstream branches.

The above algorithm suffers from some drawbacks. Firstly, the feedback implosion problem may appear because extra BRM cells are sent and no handling is presented to balance the FRM to BRM ratio. Secondly, the local congestion state is not considered. Thirdly, when receiving feedback from all branches and then passing a BRM cell, MER is reset to PCR and the M counters are decremented but not reset. This may lead to a consolidation noise problem in the next round because non-zero counters will indicate that some branches sent feedback while actually their feedback values are cleared by resetting MER. Finally, the algorithm is missing a technique to handle non-responsive branches.

Ammar et al. also proposed an enhancement to the consolidation procedure that can be applied to many proposed algorithms. In the enhancement, switches remember the previous explicit rates reported on their downstream branches. Until a new explicit rate is reported, the old value is used by the consolidation procedure. In fact, the recently-received ER information for a branch may be a good indication for the future ER from that particular branch. This enhancement overcomes common defect of clearing all stored information whenever a BRM cell is sent upstream. The enhancement needs storing all values returned by the BRM cells from all branches and updates the values as BRM cells are received. This can be an expensive operation as the number of branches increases.

3.3.8 Chen et al. Algorithm [1999]

Ammar et al. [19] proposed a probabilistic aggregation technique to modify the "wait-for-all" algorithm to improve its slow transient response. Fahmy et al. argued that an overload condition should be immediately reported to the source without the need to wait for feedback from all branches. Both algorithms use a threshold σ to determine if an extra BRM cell should be generated. However, determining the threshold is a tricky problem. The higher the threshold is, the faster the transient response is, and the higher overhead is. Chen et al. [21] designed a new algorithm to alleviate the threshold problem by providing more flexibility to span the speed-overhead spectrum.

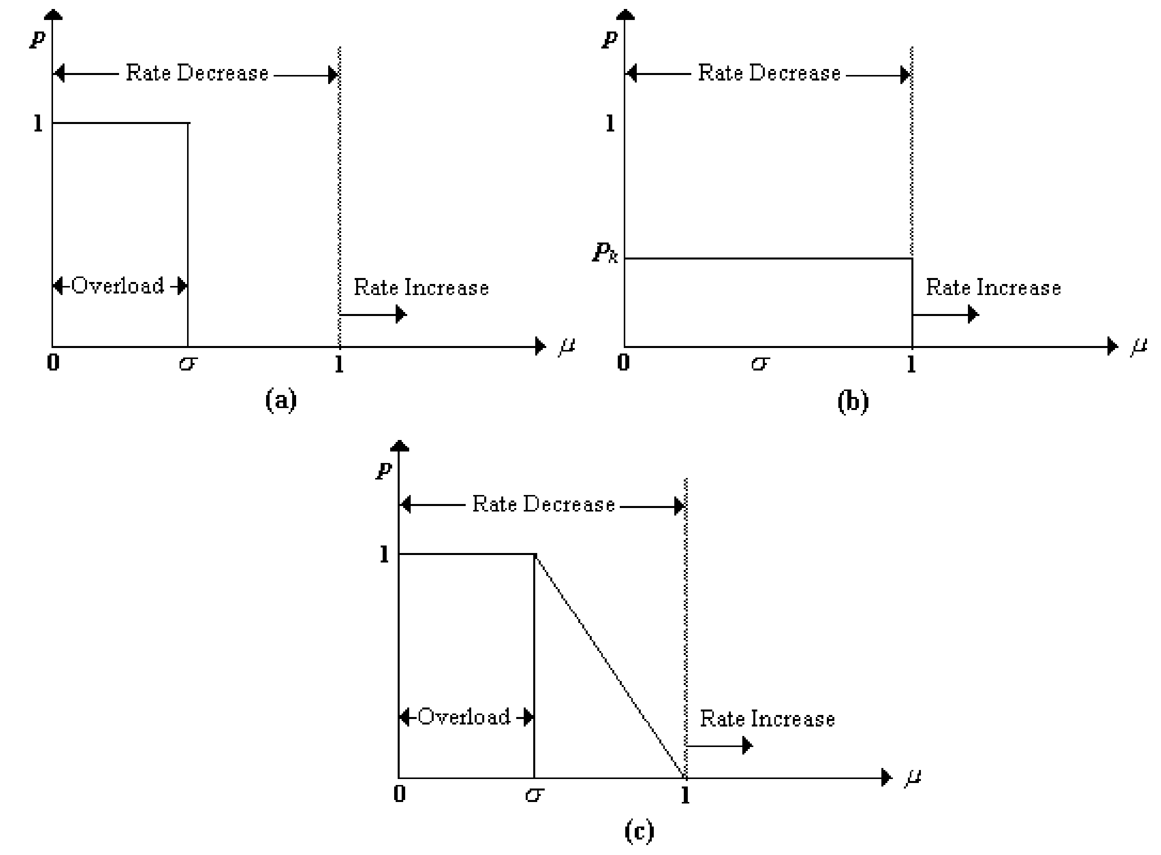

Figure 3.4: The probability p function to send an extra BRM cell:
(a) Fahmy et al. Algorithm, (b) Ammar et al. Algorithm, and (c) Chen et al. Algorithm.

Let μ denotes the ratio of the current MER and the rate indicated in the last returned BRM cell. To send an extra BRM cell, Fahmy et al. presented a step probability function to send an extra BRM cell as shown in figure 3.4(a) while Ammar et al. presented a fixed probability function with value P_k over interval (0,1) as shown in figure 3.4(b). The new scheme sends an extra BRM cell with a probability p, which is a function of the current collected MER and the last returned feedback. An extra BRM cell is sent if an overload condition is detected ($\mu < \sigma$). The probability p to send an extra BRM cell when $\sigma \le \mu \le 1$ is a linear function between two ends: one is p=1 when $\mu = \sigma$ and another is p=0 when μ=1. That is, p=(1-μ)/(1-σ) as shown in figure 3.4(c). The algorithm operates as follows.

```
Upon the receipt of an FRM cell:
Multicast FRM cell to all participating branches

Upon the receipt of a BRM cell from branch i:
1. Increase BRMRecv[i], the corresponding counter, by one
2. Let MER = min (MER, ER from BRM cell)
3. IF (all BRMRecv counters >0) THEN
       A. Pass the BRM with ER = MER to the source
       B. Reset BRMRecv counters to be zero
       C. Let MER = PCR
   ELSE IF (MER<$\sigma \times$LastER) THEN
       A. Pass the BRM with ER = MER to the source
   ELSE IF (MER<LastER) THEN
       A. Let $\mu$= MER/LastER
       B. Let $p$ = (1-$\mu$)/(1-$\sigma$)
       C. IF (RandomValue < $p$) THEN
            1. Pass this BRM cell with ER=MER to the source

When a BRM is to be scheduled:
1.Let ER=min(ER,ER calculated by rate allocation algorithm for all
branches)
2.Let LastER = ER
```

If σ is set close to 1, all three algorithms exhibit similar response since their probability functions are similar. If σ is set close to 0, Chen et al. algorithm still has a chance to send an extra BRM cell to avoid suffering from the slow transient response. Furthermore, since the algorithm sends extra BRM cells with a probability p if $\sigma \le \mu < 1$ ($0 < p < 1$), the source may decrease its rate gracefully which is beneficial to the video networking applications.

The above algorithm is missing a technique to control the BRM to FRM ratio thus it may suffer from the feedback implosion problem. The local congestion state is also not considered as a flag for extra BRM cells.

Chen et al. [21] argued that the static time-out scheme presented by Zhang et al. [19] is not appropriate for the multicast tree where the leaves are located in various distances to a branch point and the membership dynamically changes due to joining/leaving of leaves. They designed a dynamic time-out scheme [21] which also scales to very large multicast trees.

3.3.9 Ros et al. Algorithm [2000]

Ros et al. [22] proposed a new algorithm which keeps track of the M smallest available rates from the branches at each branching point. They argued that it has zero response delay, noise stability, and small probability of noise.

From Ros et al. point of view, there are two fundamental reasons for which the previous algorithms suffer from several consolidation issues

(1) The MER value has to be reset to infinity (or to the peak cell rate) every time a BRM cell is sent back to the root. Because of this, if a BRM cell is sent before all the feedback has been consolidated, noise may be generated.

(2) The only way to increase MER is to wait for it to be reset. In other words, suppose that the most bottlenecked branch increases its available rate and sends its feedback to the branching switch. Because there is no way for the switch to know that this particular feedback comes from the most bottlenecked branch, it cannot increase MER. Instead, it has to wait for a reset of MER and for a new minimization iteration to determine the new bottleneck available rate.

Ros et al. suggested that both problems can be solved if the switch stores the identifier of the branch for which it has stored the available rate. Indeed, note that (2) can be solved since now the switch can identify that the most bottlenecked branch has increased its available rate. Moreover, this provides the means to increase MER.

The previous reasoning is the motivation for the *Bottleneck Branch Marking* (BBM) algorithm. The BBM algorithm keeps track of the *M* most bottlenecked branches. For this, the switch stores a matrix with *M* entries including both the identifier of the branch (ID) and the last available rate received at the branching node. This matrix is called th*e BBM matrix* and its size *M* can be adjusted depending on the scalability and performance requirements. For example, if *N* is the number of branches, then the case *M*=*N* is equivalent to a per-branch approach. The BBM matrix has *M* entries, and each entry consist of both the ID and the available rate. A new feedback is stored in this structure only if its available rate is smaller than any of the available rates in the BBM matrix. There are two ways for which a branching switch may send a BRM cell back to the root:

(1) Whenever a new bottleneck appears: Note that this way the algorithm has zero response delay. However, that may cause the ratio BRM/FRM be bigger than one. To control this, the algorithm stores the number of extra BRM cells sent because of a new bottleneck in the *ExtraBRM* field.

(2) Whenever the number of feedback values received since the last time a BRM cell was sent is equal to the number of branches: This one is implemented with the *NBRMsRecv* field. However, if $ExtraBRM > 0$ then no BRM cell is sent. This way the algorithm ensures convergence of the ratio BRM/FRM to 1. The algorithm operates as follows.

```
Upon the receipt of an FRM cell
Multicast FRM cell to all participating branches

Upon the receipt of a BRM cell from branch i:
1. Increment NBRMsRecv
2. IF (ER from BRM cell is one of the Mth most bottleneck) THEN
       A. Store ER and i in BBM;
3. IF (ER from BRM cell is the most bottleneck) THEN
       A. Pass this BRM cell to the source
       B. IF (NBRMsRecv = Nbranches) THEN
              1. Let NBRMsRecv = 0
          ELSE
              1. Increment ExtraBRM
   ELSE
       A. IF (NBRMsRecv = Nbranches) THEN
              1. Let NBRMsRecv = 0
              2. IF (ExtraBRM = 0) THEN
                     A. Pass this BRM cell with ER= Smallest ER in BBM to
                        the source
                 ELSE
                     A. Decrement ExtraBRM
                     B. Discard this BRM cell  // to ensure BRM/FRM→1 //
          ELSE
              1. Discard this BRM cell

When a BRM is to be scheduled:
Let ER= min (ER, ER calculated by rate allocation algorithm for all
branches)
```

The trade-off between performance and complexity is reflected in the parameter M, the number of branches the switch keeps track of. This technique happens to have two nice properties. Firstly, They proved [22] that even if the number of branches *N* tends to infinity, the probability of having noise does not tend to 1. Secondly, the noise probability decreases exponentially with the number of branches *M* of which the switch keeps track. For example, they proved [22] that for $M=20$ (only the 20 most bottleneck branches are stored) and $N=\infty$ (there are infinite number of branches), the noise probability is about less than 10^{-8}. This result says that they don't need infinite storage to consolidate infinite number of branch ER values but just a small storage.

Finally, It is clear that this solution is the most expensive one. Even if just small storage is needed, it is still more than one value (MER) used by the most previous algorithms. It costs time also to maintain the BBM matrix. But the fast transient response and low consolidation noise deserve that.

The algorithm also needs to handle the non-responsive branches. It would be stuck at an old feedback of a left bottleneck leaf. The local congestion state should be also considered for that algorithm to be an ideal one.