****

ATM Forum Document Number: ATM Forum/97-0615

****

Title: Feedback consolidation algorithms for ABR point-to-multipoint connections

****

Abstract:
ABR traffic management for point-to-multipoint connections entails that the source be controlled to the minimum rate supported by all the leaves of the multicast tree. A number of algorithms have been developed for extending ABR congestion avoidance algorithms to perform the feedback consolidation operation. This contribution discusses the various design options and implementation alternatives for the consolidation algorithms, and proposes a number of new algorithms that aim at providing a faster transient response, while eliminating the noise that may be caused if the feedback is returned before all leaves have responded. The performance of the proposed algorithms is compared to the performance of the previous algorithms under a large variety of conditions. In particular, configurations with varying bottleneck locations, link lengths, traffic and background traffic types are used to demonstrate the tradeoffs among the different algorithms. The results indicate that the new algorithms we propose completely eliminate the noise, while exhibiting a very fast transient response. Hence, although the algorithms have a slightly higher implementation complexity, they offer significant performance improvement.

****

Source:
Sonia Fahmy, Raj Jain, Rohit Goyal, Bobby Vandalore, and Shivkumar Kalyanaraman
The Ohio State University (and NASA)
Department of Computer and Information Science

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

Contact: Phone: 614 292 3989; Fax: 614 292 2911; Email: {fahmy,jain}@CIS.OSU.EDU

Sastri Kota
Lockheed Martin Telecommunications
1272 Borregas Avenue
Bldg B/551 O/GB - 70
Sunnyvale, CA 94089
Email: 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
Email: psamudra@telecom.sna.samsung.com

****

Date: July 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

In point-to-point ABR flow control, the source is controlled to the minimum rate that can be supported by all the switches on the path from the source to the destination [3]. 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 path from the source to *all of the leaves* in the multicast tree. This is because the minimum rate is the technique most compatible with the typical data requirements: no data should be lost, and the network can take whatever time it requires to deliver the data intact.

The operation of feedback consolidation employed at branch points can be explained with the aid of figure 1. The consolidation operation is required to avoid the feedback implosion problem, where the number of backward RM cells received by the source is proportional to the number of leaves in the multicast tree. In addition, the source allowed cell rate (ACR) should not fluctuate due to the varying feedback received from different leaves.

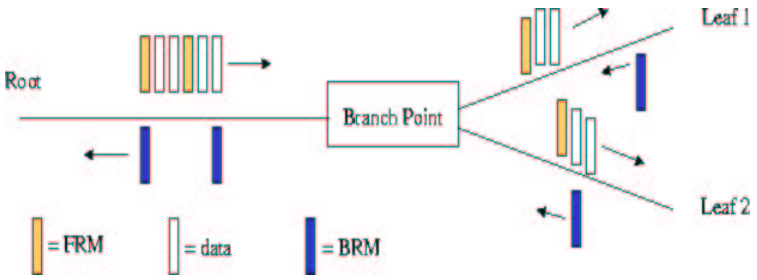


Figure 1: Point-to-multipoint connections

A number of consolidation algorithms have been proposed in [1, 12, 8, 9]. All the algorithms control the source rate to the minimum rate indicated by the backward resource management (BRM) cells. The RM cells could be turned around by the branch point [9, 12], or by the destination [1, 8]. The condition that triggers the branch point to send a BRM cell is also different in different algorithms: some of the algorithms wait for feedback to arrive from all the leaves, while some send the feedback when a forward RM (FRM) cell or a BRM cell is received.

Several design and implementation considerations come into play when developing a consolidation algorithm. There is a tradeoff between the complexity of the algorithm and its performance. Moreover, the transient response of the algorithm should not be sacrificed to reduce the noise occurring when feedback from some of the leaves is not incorporated into the BRM cell returned. Finally, the scheme must be scalable to very large multicast trees. The implementation complexity, feedback delay, and the overhead of the backward RM cells, should not increase with the increase of the number of levels or leaves of the multicast tree.

In this contribution, we propose a set of consolidation algorithms that aim at providing a fast transient response, while eliminating consolidation noise. We examine the performance of the proposed algorithms and compare it to the previously proposed ones in complexity, transient response, consolidation noise, and scalability. The remainder of this contribution is organized as follows. The next section summarizes the previous work on point-to-multipoint ABR flow control. A discussion of the various design and implementation issues involved is presented next, followed by a detailed description and pseudocode of the previously proposed algorithms, and the new ones we propose.

All the algorithms are then simulated and their performance is analyzed under a variety of configurations. The contribution concludes with a detailed discussion of the performance and complexity tradeoffs among the different algorithms.

2 Background and Related Work

In [9], a simple point-to-multipoint ABR framework was proposed where the source in a point-to-multipoint virtual connection (VC) sends at the minimum of the rates allowed by all the destination nodes. The algorithm works as follows. A register to store the minimum explicit rate feedback, MER, is set to the minimum of its current value and the ER in the BRM cell whenever a BRM cell is received from one of the branches. When an FRM cell is received, it is multicast to all branches, and a BRM is returned with the MER value as the ER. MER is then set to the ER value in the FRM cell (typically PCR). Thus the minimum of the rates supported by the branches is returned to the source [9, 10, 11]. However, only those branches that returned BRM cells since the last BRM cell was sent are represented in the feedback.

Several variations on the previously described algorithm were proposed in [8]. Some of the new schemes are simpler to implement than the previous proposal that required the branch point to generate a returning RM cell for every forward RM cell. Other schemes attempt to achieve better performance.

The early proposal suffers from the "consolidation noise" problem, where a BRM generated by a branch point may not consolidate feedback from all tree branches [4]. In fact, if a BRM generated by a branch point does not accumulate feedback from *any* branch, the feedback can erroneously be given as the peak cell rate (if that branch point itself is not overloaded). A simple enhancement to alleviate this problem is to maintain a flag, and only generate the BRM cell if a BRM has been received from a leaf since the last BRM was sent by the branch point [8, 12].

Another idea proposed in [8] reduces the complexity of the algorithm as follows. The backward RM cells are generated by the destinations and *not* by the branch points, which is similar to the case of unicast [7]. The motivation behind this modification is as follows. If branch points turn around RM cells, the implementation may have a high cost. In the previously described algorithms, the number of BRMs generated by branch points per forward RM cell from the source is proportional to the number of branch points in a multicast tree. The new algorithm does not generate BRM cells at branch points whenever FRM cells are received, but simply sets a flag indicating the receipt of the FRM cell, and multicasts it to all leaves. When a BRM cell is received from a branch, it is passed back to the source (after using the minimum allocation), only if the flag was set. The flag is then reset, and the MER value set to the peak cell rate [8].

It is natural to extend this idea to only send back the BRM cell when BRM cells from *all* branches are received. This can be easily implemented by maintaining a separate bit for each branch that indicates if a BRM cell has been received since the last BRM cell was sent. Clearly this method incurs additional complexity, compared to the previous one. Moreover, it has to deal with the problems of failure of one of the branches by implementing timeouts. The four variations of the algorithm were compared in [8].

A similar method to the one last described was proposed in [1]. Again, the algorithm only allows

feedback to return to the source when BRM cells have been received from all branches. However, the scheme proposes to add a sequence number to the RM cells. The BRM cell that is allowed to pass back to the source is the last BRM cell to be received with a certain sequence number. This guarantees that among all BRM cells with the same sequence number, one and only one BRM cell passes back to the source.

3 Design and Implementation Issues

As pointed out in the previous section, there are several different ways to implement the point-to-multipoint consolidation algorithms. Each method offers a tradeoff in complexity, scalability, overhead, transient response, and consolidation noise. In addition, several guidelines and metrics can be used to evaluate the performance of the algorithms. The tradeoffs and issues can be summarized as follows:

- Which component generates the BRM cells (i.e. turns around the FRM cells)? Should the branch point or the destination perform that operation? If the branch point turns around the BRM cells, the scheme may incur more overhead.
- Should the branch point wait for feedback from all the branches before passing the BRM cell upstream? Although this eliminates the consolidation noise problem, it incurs additional overhead, and increases the transient response of the scheme, especially after idle or low rate periods. This can have severe implications in case of sudden overload, since the queues can substantially grow while the branch point is still waiting for feedback from distant leaves.
- How can the ratio of FRM cells generated by the source to BRM cells returned to the source be controlled? Some algorithms send a BRM cell when an FRM cell is received, when an FRM cell is received after a BRM cell has been received, or when a BRM cell is received after an FRM cell has been received. This guarantees that the number of BRM cells is less than or equal to the number of FRM cells. If the scheme waits for feedback from all leaves, the BRM to FRM ratio is guaranteed to be less than or equal to one if all the leaves are responsive. What if the scheme does not wait for feedback from all leaves in the case of overload? The ratio must be controlled by other means in this case.
- A related issue is the ratio of BRM cells *in the network* to the number of source-generated FRM cells. Algorithms that turn around RM cells at the branch points increase the number of BRM cells inside the network. The number becomes dependent on the number of branch points and the number of levels of the multicast tree. This has adverse effects on the overhead and the scalability of the scheme.
- Another issue is the operation of the branch point when the branch point is also a switch and queuing point. In this case, the coupling of the switch and the branch point functions must be considered: When should the actual rate computation algorithm be performed? The point-to-point algorithm should simply be a special case of the point-to-multipoint algorithm, when the number of branches is equal to one. In our experiments in this contribution, the ERICA switch algorithm [6, 5] is executed just before the BRM cell is sent. Thus the algorithm is performed only once per BRM cell returning to the source (and not per-BRM cell arriving

at the destination). This makes the point-to-multipoint algorithm scalable to large numbers of leaves or branch points. When the BRM cell is to be sent out, the ERICA algorithm has to be performed for each of the branches to account for overload on each of the outgoing ports, and the minimum of all these values should be indicated in the returning RM cell. But what happens when there is sudden overload at this switch and the branch point is waiting for BRM cells from all leaves? Should the branch point invoke ERICA and if overload is detected, not wait for all branches?

- Some algorithms wait for an FRM cell to be received to send feedback. What are the implications of this on the scalability of the scheme? Will the feedback delay grow with the number of branches? For example, figure 2 illustrates a configuration where there are two branch points on the path from the source to one of the leaves of the multicast tree. If you have to wait for the next FRM cell at each of the branches, the time to return a BRM cell can increase with the number of levels of the multicast 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 last branch do not suffer from this problem, so they are less sensitive to number of branch points.

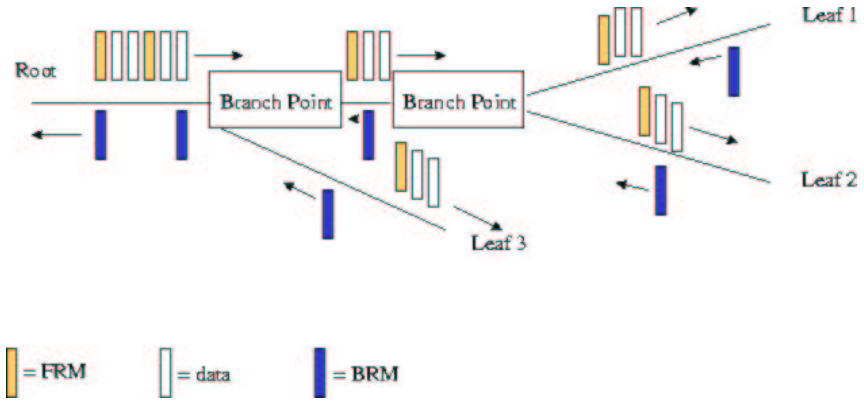


Figure 2: Multiple branch points

- All the consolidation algorithms use some registers at the branch point to store values such as the minimum rate given by all the leaves in the current iteration, flags to indicate whether an RM cell has been received since the last one was sent, and many others. It is important to note that values such as minimum explicit rate, the flags, the number of BRM cells received, should not be stored per output port like all the other metrics that point-to-point algorithms are using for rate calculation. In fact, values such as the CCR (or ACR in case the branch point is a source) should not be stored per output port either, since it is the same for all branches. Hence the CCR, minimum values and flags should be stored only once per point-to-multipoint VC. However, the flags indicating that a BRM cell has been received from a certain branch and values related to the timeout algorithm (described in the next issue) should be stored per output port, like the other port-related values (for example, overload factor, target capacity, input rate, number of active sources).

- If the consolidation scheme waits for feedback from *all* the branches before sending it to the source, an algorithm must be developed to determine when a branch becomes non-responsive and handle such non-responsive branches. If a branch becomes non-responsive, then sending

feedback should not be delayed indefinitely waiting for feedback from that non-responsive branch. Clearly, a timeout mechanism must be implemented. Thus, a timeout value must be associated with each branch. The timeout value depends on the round trip time from the branch point to the leaf. This value must be continuously measured and updated, since the queuing delay of the switches on the path from the branch point to the leaf (and in the opposite direction) is a continuously changing quantity. When the timeout value for a branch is exceeded, the branch is declared to be in the non-responding state, and the branch point does not account for that branch in its feedback information, until the branch returns to the responding state.

These issues are discussed further is later sections.

4 Feedback Consolidation Algorithms

In the four algorithms presented next, the ERICA algorithm [5, 6] is employed immediately before sending a BRM on the link. This ensures that the most recent feedback information is sent. The algorithms at the branch point operate as explained in the following subsections.

4.1 Algorithm 1

This is a modified version of the algorithm presented in [9]. A register MER, and two flags MCI and MNI are maintained for each multipoint VC. The variables store the minimum of the explicit rate (ER), congestion indication (CI) and no increase (NI) indicated in the BRM cells which were received after the last BRM cell was sent. Three temporary variables: MXER, MXCI, and MXNI are also used when an FRM cell is received (their values do not persist across invocations of the algorithm). They store the ER, CI and NI 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, MXCI = CI from FRM cell, MXNI = NI from FRM cell
3. Return a BRM with ER = MER, CI = MCI, NI = MNI to the source
4. Let MER = MXER, MCI = MXCI, MNI = MXNI

Upon the receipt of a BRM cell:

1. Let MER = min (MER, ER from BRM cell), MCI = MCI OR CI from BRM cell, MNI = MNI OR NI from BRM cell
2. Discard the BRM cell

When a BRM is about to be scheduled:
Let ER = min (ER, ER calculated by ERICA for all branches)

4.2 Algorithm 2

This is a modified version of the second algorithm in [8]. The only change from Algorithm 1 is ensuring that at least one BRM cell has been received from a leaf before turning around an FRM. For this purpose, a boolean flag, AtLeastOneBRM is set to true when a BRM cell is received from a branch, and it is reset when a BRM is sent by the branch point. As before, MER, MCI, MNI, and here, AtLeastOneBRM, are stored for each multipoint VC. Again, MXER, MXCI, MXNI are just temporary variables.

Upon the receipt of an FRM cell:

1. Multicast FRM cell to all participating branches
2. IF AtLeastOneBRM THEN
   - Let MXER = ER from FRM cell, MXCI = CI from FRM cell, MXNI = NI from FRM cell
   - Return a BRM with ER = MER, CI = MCI, NI = MNI to the source
   - Let MER = MXER, MCI = MXCI, MNI = MXNI
   - Let AtLeastOneBRM = 0

Upon the receipt of a BRM cell:

1. Let AtLeastOneBRM = 1
2. Let MER = min (MER, ER from BRM cell), MCI = MCI OR CI from BRM cell, MNI = MNI OR NI from BRM cell
3. Discard the BRM cell

When a BRM is about to be scheduled:
Let ER = min (ER, ER calculated by ERICA for all branches)

4.3 Algorithm 3

This is a modified version of the third algorithm in [8]. The main idea here is that the branch point does not turn around the FRMs, but the BRM that is received from a leaf immediately after an FRM has been received by the branch point is passed back to the source, with the minimum values. A boolean flag, AtLeastOneFRM, indicates that an FRM cell has been received by the branch point after the last BRM cell was passed to the source. Again, MER, MCI, MNI, and AtLeastOneFRM are stored per multipoint VC.

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), MCI = MCI OR CI from BRM cell, MNI = MNI OR NI from BRM cell
2. IF AtLeastOneFRM THEN
   - Pass the BRM with ER = MER, CI = MCI, NI = MNI to the source
   - Let MER = PCR, MCI = 0, MNI = 0
   - Let AtLeastOneFRM = 0

   ELSE Discard the BRM cell

When a BRM is about to be scheduled:
Let ER = min (ER, ER calculated by ERICA for all branches)

4.4 Algorithm 4

A variation of this algorithm was presented in [8] as algorithm 4, and another variation using sequence numbers in RM cells was proposed in [1]. The main idea here is that a BRM is passed to the source only when BRM cells have been received from *all* branches. 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, NumberOfBRMsReceived is incremented the first time a BRM cell is received from each branch. As before, the MER, MCI, MNI, and NumberOfBRMsReceived registers are maintained per multipoint VC. The value of the NumberOfBRMsReceived counter is compared to the value of another counter, NumberOfBranches, every time a BRM cell is received by the branch point. NumberOfBranches stores the number of branches of the point-to-multipoint VC at this branch point. The register NumberOfBranches is also stored for each VC, and is initialized during connection setup. In addition, if leaf initiated join is allowed (as in UNI 4.0), NumberOfBranches must be updated every time a branch is added to a branch point. If the value of NumberOfBRMsReceived is equal to NumberOfBranches, the BRM cell is passed back to the source, carrying the values of the MER, MCI and MNI registers.

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, since it is needed for each branch.

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

Upon the receipt of a BRM cell:

1. IF NOT BRMReceived THEN
   Let BRMReceived = 1
   Let NumberOfBRMsReceived = NumberOfBRMsReceived + 1
2. Let MER = min (MER, ER from BRM cell), MCI = MCI OR CI from BRM cell, MNI = MNI OR NI from BRM cell

3. IF NumberOfBRMsReceived is equal to NumberOfBranches THEN

   - Pass the BRM with ER = MER, CI = MCI, NI = MNI to the source
   - Let MER = PCR, MCI = 0, MNI = 0
   - Let NumberOfBRMsReceived = 0
   - Let BRMReceived = 0 FOR all branches

   ELSE Discard the BRM cell

When a BRM is about to be scheduled:
Let ER = min (ER, ER calculated by ERICA for all branches)

Note that a timeout mechanism must be implemented to ensure that BRM cell flow is not stopped in the case of non-responding branches.

5 New Algorithms

The main problem with algorithm 4 described in the previous section is its slow transient response. Even when high overload has been 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 might have transmitted a large number of cells which would be dropped due to buffer overflow, leading to 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.

The main idea behind the algorithms presented next is that the slow transient response problem should be avoided when an overload situation has been detected. In this case, there is no need to wait for feedback from all the leaves, and the overload should be immediately indicated to the source. In cases of underload indication from a leaf, it is better to wait for feedback from all the leaves, since another branch 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 when the feedback to be indicated is *much less* than the last feedback returned by this branch point (the "much less" condition can be tested using an additive or multiplicative value or factor). An alternative method would be to compare the feedback to be indicated to the *current cell rate (CCR)* or ACR of the VC. Although this may be better because it accounts for upstream bottlenecks, and prevents the transmission of unnecessary BRM cells in such cases, the CCR information may be stale due to the delay from the source to the branch point (it may also be much larger when the source becomes idle or becomes a low rate source after the last FRM was sent), and a large number of BRMs may be sent in such cases. The last feedback indicated by the branch point is a more current value. The minimum of the CCR and last feedback given can be used in the comparison, but this involves some additional complexity, and may slow down the overload response when the CCR happens to have been small, but is currently large.

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

5.1 Fast Overload Indication (Algorithm 5)

In this algorithm, the LastER register maintains the last explicit rate value returned by this branch point. The registers MER, MCI, MNI, NumberOfBRMsReceived, NumberOfBranches and LastER, are stored per multipoint VC. As before, BRMReceived is stored per output port.

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 not true, only if a BRM cell is used to indicate overload conditions, and hence the counters and register values should not be reset. FRMminusBRM is not a real variable; it is only used for accounting purposes, and will not exist in a real implementation.

Upon the receipt of an FRM cell:

1. Multicast FRM cell to all participating branches
2. Let FRMminusBRM = FRMminusBRM + 1

Upon the receipt of a BRM cell:

1. Let SendBRM = 0
2. Let Reset = 1
3. IF NOT BRMReceived THEN
   Let BRMReceived = 1
   Let NumberOfBRMsReceived = NumberOfBRMsReceived + 1
4. Let MER = min (MER, ER from BRM cell), MCI = MCI OR CI from BRM cell, MNI = MNI OR NI from BRM cell
5. IF MER << LastER THEN (* overload is detected *)
   IF NumberOfBRMsReceived < NumberOfBranches THEN
   Let Reset = 0
   Let SendBRM = 1
   ELSE IF NumberOfBRMsReceived is equal to NumberOfBranches THEN
   Let SendBRM = 1
6. IF SendBRM THEN
   - Pass the BRM with ER = MER, CI = MCI, NI = MNI to the source
   - IF Reset THEN
     Let MER = PCR, MCI = 0, MNI = 0
     Let NumberOfBRMsReceived = 0
     Let BRMReceived = 0 FOR all branches
   - Let FRMminusBRM = FRMminusBRM − 1

   ELSE Discard the BRM cell

When a BRM is about to be scheduled:

1. Let ER = min (ER, ER calculated by ERICA for all branches)
2. Let LastER = ER

5.2 RM Ratio Control Option (Algorithm 6)

The previous algorithm may increase the BRM cell overhead, since the ratio of source-generated FRM cells to BRM cells received by the source can be more than one. To avoid this problem, we introduce the register SkipIncrease which is maintained for each multipoint VC, and initialized to zero. 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 more than zero, this particular feedback can be ignored and SkipIncrease is decremented. Note that the value of the SkipIncrease counter will not increase to large values, since the congestion avoidance algorithm (such as ERICA) arrives at the optimal allocation within very few iterations, and the rate allocations cannot continue decreasing indefinitely.

A problem with this approach, though, is that the BRM cells at a branch point may be exhausted for some time, and hence new ones discarded in underload situations, even when there is sudden overload upstream, and the BRMs are needed. However, as seen in the performance analysis in the next sections, this situation rarely arises since few BRMs are actually discarded, and SkipIncrease always has small values. Observe that this situation can be alleviated if the CCR value (and not the last ER returned) was being compared to the MER (since the CCR contains information about the feedback given from upstream switches).

Upon the receipt of an FRM cell:

1. Multicast FRM cell to all participating branches
2. Let FRMminusBRM = FRMminusBRM + 1

Upon the receipt of a BRM cell:

1. Let SendBRM = 0
2. Let Reset = 1
3. IF NOT BRMReceived THEN
   Let BRMReceived = 1
   Let NumberOfBRMsReceived = NumberOfBRMsReceived + 1
4. Let MER = min (MER, ER from BRM cell), MCI = MCI OR CI from BRM cell, MNI = MNI OR NI from BRM cell
5. IF MER $\geq$ LastER AND SkipIncrease $>$ 0 AND NumberOfBRMsReceived is equal to NumberOfBranches THEN

Let SkipIncrease = SkipIncrease − 1
Let NumberOfBRMsReceived = 0
Let BRMReceived = 0 FOR all branches
ELSE IF MER << LastER THEN
IF NumberOfBRMsReceived < NumberOfBranches THEN
Let SkipIncrease = SkipIncrease + 1
Let Reset = 0
Let SendBRM = 1
ELSE IF NumberOfBRMsReceived is equal to NumberOfBranches THEN
Let SendBRM = 1

6. IF SendBRM THEN
   - Pass the BRM with ER = MER, CI = MCI, NI = MNI to the source
   - IF Reset THEN
     Let MER = PCR, MCI = 0, MNI = 0
     Let NumberOfBRMsReceived = 0
     Let BRMReceived = 0 FOR all branches
   - Let FRMminusBRM = FRMminusBRM − 1

   ELSE Discard the BRM cell

When a BRM is about to be scheduled:

1. Let ER = min (ER, ER calculated by ERICA for all branches)
2. Let LastER = ER

5.3 Immediate Rate Calculation Option (Algorithm 7)

The previously discussed algorithms can offer very fast congestion relief when an overload is detected in a branch of the multicast tree. However, they do not account for the potential overload situation at the branch point itself, since if the branch point is a switch, the ERICA 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 a queuing point, the immediate rate calculation option invokes ERICA whenever a BRM is received, and not just when a BRM will be sent. Hence overload at the branch point can be detected and indicated according to the fast overload indication option as previously described. However, this option involves some additional complexity.

The algorithm presented next is the same as Algorithm 6 in the previous subsection, except for the addition of the ERICA invocation (italicized below).

Upon the receipt of an FRM cell:

1. Multicast FRM cell to all participating branches

2. Let FRMminusBRM = FRMminusBRM + 1

Upon the receipt of a BRM cell:

1. Let SendBRM = 0
2. Let Reset = 1
3. IF NOT BRMReceived THEN
   Let BRMReceived = 1
   Let NumberOfBRMsReceived = NumberOfBRMsReceived + 1
4. Let MER = min (MER, ER from BRM cell), MCI = MCI OR CI from BRM cell, MNI = MNI OR NI from BRM cell
5. *Let MER = minimum ER calculated by ERICA for all branches*
6. IF MER $\geq$ LastER AND SkipIncrease $>$ 0 AND NumberOfBRMsReceived is equal to NumberOfBranches THEN
   Let SkipIncrease = SkipIncrease $-$ 1
   Let NumberOfBRMsReceived = 0
   Let BRMReceived = 0 FOR all branches
   ELSE IF MER $<<$ LastER THEN
   IF NumberOfBRMsReceived $<$ NumberOfBranches THEN
   Let SkipIncrease = SkipIncrease + 1
   Let Reset = 0
   Let SendBRM = 1
   ELSE IF NumberOfBRMsReceived is equal to NumberOfBranches THEN
   Let SendBRM = 1
7. IF SendBRM THEN
   - Pass the BRM with ER = MER, CI = MCI, NI = MNI to the source
   - IF Reset THEN
     Let MER = PCR, MCI = 0, MNI = 0
     Let NumberOfBRMsReceived = 0
     Let BRMReceived = 0 FOR all branches
   - Let FRMminusBRM = FRMminusBRM $-$ 1

   ELSE Discard the BRM cell

When a BRM is about to be scheduled:

1. Let ER = min (ER, ER calculated by ERICA for all branches)
2. Let LastER = ER

6 Performance Analysis

This section provides a performance comparison among all the consolidation algorithms, in a variety of configurations with bursty and non-bursty traffic, with and without VBR, and with various link lengths, bottleneck locations, and number of leaves. A large number of other configurations was also tested (see [2, 5] for some of the configurations), but only a sample of the results is shown here. In particular, configurations with a large number of leaves at varying distances in the multicast tree were simulated, and the results were consistent with the other results.

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

- (a) Graph of allowed cell rate (ACR) in Mbps over time for each source
- (b) Graph of ABR queue lengths in cells over time at each switch

6.1 Parameter Settings

Throughout our experiments, the following parameter values are used:

1. All links have a bandwidth of 155.52 Mbps (149.76 Mbps when SONET overhead is accounted for).
2. Except where indicated, all links are 1000 kms in length.
3. All point-to-multipoint traffic flows from the root to the leaves of the tree. No traffic flows from the leaves to the root, except for RM cells. The same applies for point-to-point connections.
4. The source parameter Rate Increase Factor (RIF) is set to one, to allow immediate use of the full explicit rate indicated in the returning RM cells at the source. ICR is also set to a high value (almost Peak Cell Rate). These factors are set to such high values to simulate a worst case load situation.
5. 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 from the rate reductions due to TBE.
6. The switch target utilization parameter was set at 90%.
7. The switch averaging interval was set to the minimum of the time to receive 100 cells and 1 ms (the recommended value for WANs).
8. All sources are deterministic, i.e., their start/stop times and their transmission rates are known. The bursty traffic sources send data in bursts, where each burst starts after a request has been received from the client.
9. VBR sources are on for 20 ms and off for 20 ms.
10. Simulation time ranges from 200 ms to 800 ms to allow the system enough time to reach steady state.

6.2 Simulation Results

This section discusses the performance of the consolidation algorithms by comparing them in a set of configurations. Figures 6 through 12 illustrate the performance of the seven different algorithms in a situation where there is both variable capacity and variable demand. These situations offer the toughest challenge for rate allocation algorithms [5, 2]. The configuration simulated is shown in figure 3. The source indicated by W is a bursty source, I is a persistent (infinite) source, while V is a VBR source.

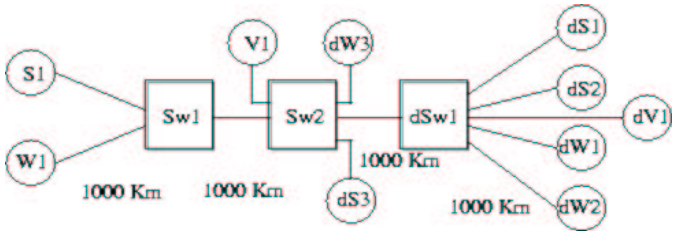


Figure 3: WAN parking lot configuration with bursty, infinite and VBR connections

The rate graphs for algorithms 1, 2, and 3 show a lot of fluctuations and inaccurate (around 140 Mbps) feedback given in the initial 150 ms. This leads to large queues. Algorithm 4 gives more accurate feedback, but the feedback is given after around 50 ms, which results in large initial queues. Algorithms 5 and 6 produce identical results to algorithm 4, since the bottleneck link is attached to the branch point. Algorithm 7, on the other hand, exhibits a fast transient response, and gives relatively accurate feedback to both sources. Hence, it offers the best performance since it combines the benefits of algorithm 4 with a fast transient response.

Figures 13 through 19 show the situation for a similar configuration. The configuration is the same as the previous one (figure 3), but, instead of a bursty source and an infinte one, there are 2 infinte sources (source name W is retained to distinguish the 2 connections, but it is also a persistent source).

It is clear that all algorithms exhibit less fluctuations since the demand is not variable in this configuration. However, figure 16 illustrates how the slow transient response of algorithm 4 again leads to large queues. Algorithms 5 and 6 are again similar since the bottleneck link is again attached to the branch point. It is clear that algorithm 7 avoids those problems due to its fast transient response, and hence, the queue lengths are much smaller in this case. Also note that the high initial cell rate (ICR) and rate increase factor (RIF) [3] values are the reason for the unusually large queues.

The chain configuration, illustrated in figure 4 consists of a point-to-multipoint connection where one of the links on the route to the farthest leaf is the bottleneck link. Also the link lenghts are increasing by an order of magnitude in the last two hops.

As seen in figures 20 through 26, this configuration is an ideal configuration for illustrating the consolidation noise problem. The problem is severe for algorithms 1, 2 and 3 (especially 3) (see figures 20 through 22), and leads to rate oscillations, instability, unbounded queues and unfairness against source S4 whose rate remains at half of the bandwidth, while the rate of S1 continues to oscillate around a mean of about 103 Mbps. Although using a scheme such as ERICA+ leads to stability and bounded queues in this case, the persistent rate oscillations lead to unacceptable performance

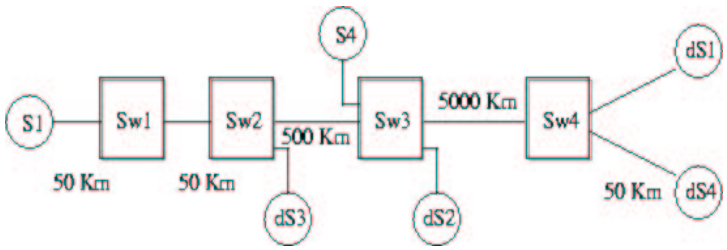


Figure 4: WAN Chain configuration

and unfairness. Algorithms 4, 5 and 6 (figures 23 through 25) avoid the noise completely, but suffer from a slow transient response. The rate of the source S1 only drops after around 60 ms, and by that time, large queues have built up at the switches). Algorithm 7 yields optimal performance in this case, since the rate of the source S1 immediately drops to its optimal value, as soon as the overload is detected.

Observe that algorithms 5 and 6 also yield near optimal performance (as algorithm 7) if the destination dS3 was further than dS1 (see the configuration in figure 5), as illustrated in the figures 27 through 33. In figures 27 through 33, the configuration simulated is modified such that the bottleneck link is closer to the branch point at switch Sw2 than another leaf, namely dS3, as in figure 5.

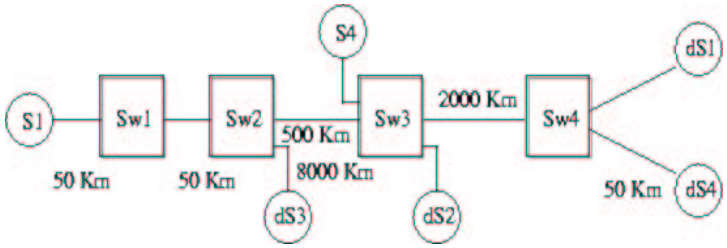


Figure 5: Modified chain configuration

In this case, as seen in figure 30, algorithm 4 wastes a long time waiting for feedback from dS3, while it has already received the bottleneck feedback from Sw3. In this case, algorithms 5, 6, and 7 send the feedback as soon as the overload situation is indicated by the BRM cell coming from switch Sw3, and do not needlessly wait for the BRM from dS3. Hence, the 3 algorithms perform near optimally since the rate of the source S1 goes to the optimal value after only around 20 ms for algorithms 5 and 6, and less than 10 ms for algorithm 7. The maximum queue lengths are also much smaller than for algorithm 4 ($>$ 16000 cells): for algorithms 5 and 6, they are around 7000 cells and for algorithm 7, they are only 3500 cells.

We have observed a similar, but more pronounced behavior when we simulated configurations with a larger number of leaves at varying distances and at varying levels of the multicast tree. The situation was much worse there with algorithms 1, 2, and 3, which had much more severe noise problems. Algorithm 4 had an *extremely* slow transient response, while algorithms 5, 6, and *especially 7* exhibited fast rate changes to the optimal values, and small queues at the switches.

7 Comparison and Interoperability of the Algorithms

This section summarizes the main conclusions from the comparison of the various algorithms. All the algorithms preserve the fairness and efficiency of the point-to-point congestion avoidance algorithm employed. We compare the space and time complexity, transient response, consolidation noise, algorithm overhead and scalability, and discuss the interoperability of various algorithms.

7.1 Implementation Complexity

Algorithms 1 and 2 are complex to implement because the branch point has to turn around the RM cells. This is somewhat similar to the Virtual Source/Virtual Destination (VS/VD) concept. Most studies argue that turning around RM cells has a high implementation complexity.

Algorithm 3 is definitely the simplest algorithm to implement, since it does not turn around RM cells, and it keeps minimal per-VC accounting information. Algorithm 4 is more complex since it has to maintain the number of branches and the number of branches from which BRMs have been received and compare those numbers. In addition, it has to maintain a bit for each output port to denote whether a BRM cell has been received from this branch, and some timeout-related values.

Algorithms 5 and 6 are slightly more complex since they may also store the last ER sent by the branch point. Alternatively, they can use the CCR of the source, which is already stored and used by most congestion avoidance algorithms (it is used in the ERICA algorithm which we have employed in this study). The additional complexity mainly stems from the comparison of the MER value to the last ER sent or the CCR value, and maintaining the SkipIncrease counter. An additional comparison and integer register do not incur much overhead though.

Algorithm 7 is somewhat more complex than algorithms 5 and 6, since it invokes the ERICA algorithm for all the branches whenever a BRM cell is received, and not only when a BRM cell is to be sent.

7.2 Transient Response

Algorithm 1 exhibits a very fast transient response. Algorithms 2 and 3 also have a reasonable transient response, since, even if there are no RM cells in the network, the feedback is quickly returned on the first BRM arrival.

Algorithm 4 has a slow transient response, since it waits for feedback from all the leaves before sending BRMs. This is especially severe in cases when there are few or no RM cells already in the network, such as during startup periods and for bursty sources. Therefore feedback can be delayed up to a function of the longest round trip times of the leaves. Algorithms 5, 6 and 7 tackle this problem for overload situations. The transient response of the schemes is very fast when an overload is detected downstream (for algorithms 5 and 6), or at this branch and downstream (for algorithm 7). In such cases, the transient response of the scheme is reasonably fast, and potential cell loss and retransmissions are alleviated.

7.3 Consolidation Noise

Algorithms 1, 2, and 3 suffer from severe consolidation noise problems. In particular, algorithms 1 and 3 (especially 3) suffer from unacceptable consolidation noise in some cases (recall figures 20 and 22). Algorithm 2 somewhat alleviates these problems, since BRMs are not sent if no feedback has been received from any of the downstream components. However, it still exhibits considerable noise.

Algorithms 4, 5, 6, and 7 eliminate this problem by waiting for feedback from all branches. Although algorithms 5, 6, and 7 do not wait for feedback from all leaves in cases of overload, this *does not* introduce noise, since the RM cells that are sent faster than the usual cells carry overload information, which would have been conveyed by the next minimum value anyway.

7.4 Scalability Issues

Algorithms should be scalable in the sense that their overhead and feedback delay should not grow with the growth of the number of branch points or levels of the multicast tree:

- **RM cell overhead:** The number of FRM cells generated by the source and the number of BRM cells received by the source should be approximately the same. Algorithm 1 generates a BRM cell at the branch point for every FRM cell it receives, thereby guaranteeing that the BRM to FRM ratio remains one. Algorithms 2 and 3 maintain a BRM to FRM ratio of less than or equal to one as follows. Algorithm 2 generates a BRM for an FRM only if a BRM has been received from a leaf since the last BRM was sent by the branch point. Algorithm 3 allows a BRM to pass to the source only if an FRM cell has been received by the branch point after the last BRM cell was forwarded by the branch point. Therefore both algorithms maintain a ratio that is less than or equal to one (actually, it is strictly less than one for algorithm 2, since the first FRM cell will never be turned).

  Algorithm 4 also maintains a ratio close to one, since one BRM cell is returned when BRM cells have been received from all branches. Algorithm 5 does not guarantee that the ratio remains at one, since RM cells carrying overload indication are allowed to quickly return to the source. Algorithms 6 and 7 fix this problem by maintaining a counter that is incremented for every extra RM cell passed, and then decremented (and the BRM cell is discarded) in cases of RM cells carrying underload information, when that counter exceeds zero. Hence, over the long run, the ratio is maintained at one. Clearly, the counter cannot increase indefinitely, since the rates cannot decrease indefinitely. In all cases we have examined, the counter did not increase beyond a small value, since the ERICA algorithm quickly arrives at the optimal allocation and exhibits a fast transient response.

  In addition to the BRM to FRM ratio at the source, the number of BRM cells generated *in the network* per source-generated FRM cell should be controlled. In algorithms 1 and 2, the switch turns around the FRM cells and produces BRM cells, but the same FRM cells are multicast to other branch point and to the leaves, and these also turn around the FRMs. Hence, the number of BRMs in the network can grow with the increase of the number of branch points. This is extremely undesirable. Algorithms 3, 4, 5, 6, and 7 solve this problem, since switches do not turn around the FRM cells.

- **Sensitivity to the maximum number of branch points on a path (levels of the tree):** Algorithm 1 waits for an FRM cell to arrive before it can send the feedback information it has consolidated from the BRM cells. This has to be done at every branch point, leading to a delay that increases with the number of levels of the multicast tree. Algorithm 2 suffers from the same drawback, since the algorithm also sends a BRM cell at the branch point when an FRM cell is received.

  Algorithm 3 is less sensitive to the number of levels of the multicast tree (contrary to what [1] argues). The BRM cell is passed to the source only if an FRM cell has been received since the last BRM cell was sent by the branch point. However, it is passed without additional delay.

  Algorithms 4, 5, 6, and 7 also do not suffer from high sensitivity to the multicast tree levels since the delay (the time between the transmission of the FRM cell at the source until the source receives the corresponding BRM cell) is only dependent on the round trip times from the source to the leaves at that particular time. Observe that the round trip times to the leaves can vary with time, dependent on the queuing delay of the switches on the path of the multicast tree. Also note that more than one leaf can have an effect on that delay since BRM cells arrive asynchronously at the branch points. However, is independent of the number of levels or number of branch points of the multicast tree.

7.5 Interoperability of the Algorithms

The various consolidation algorithms should be able to interoperate with each other if no one algorithm is standardized. It is necessary to examine how the algorithms will work together. It seems that all the algorithms can interoperate smoothly with each other, but the performance of a network with different algorithms at the different branch points, and point-to-multipoint VCs that branch at several branch points with different algorithms will need further study if a consolidation algorithm is not standardized.

8 Summary and Conclusions

Table 1 shows a summary of the results of the comparison between the consolidation algorithms. In terms of complexity, algorithm 3 is clearly the simplest. Algorithms 1 and 2 are simple, except that the RM turn around operation is expensive. Algorithm 4 introduces additional complexity to algorithm 3, since it maintains per-branch variables and performs comparisons. Algorithm 5 introduces a little more complexity to 4; algorithm 6 introduces a little more to 5, and algorithm 7 introduces some more to 6, but most of the increments are of small complexity.

The transient response of algorithm 1 is fast, but can be erroneous. Algorithms 2 and 3 offer medium response, while algorithm 4 is clearly slow. Algorithms 5, 6, and especially 7, have a fast response when overload is detected.

Consolidation noise is a problem with algorithms 1, 2, and 3, especially 1 and 3. The other algorithms overcome that problem.

As for RM cell overhead, the ratio of BRM cells received by the source to FRM cells sent by the

source is maintained at unity by algorithm 1. It is less than one for algorithm 2 (at least the first FRM is not returned), and is less than or equal to one for algorithm 3. Algorithm 4 ensures the ratio is maintained close to one. Algorithm 5 introduces additional BRM cells in case of overload, while algorithms 6 and 7 ensure the ratio is one over the long run (lim in the table means the limit as time goes to infinity).

Finally, the sensitivity of algorithms 1 and 2 to the number of branch points and the levels of the multicast tree is high due to the additional delay waiting for an FRM cell at each branch point, and the additional BRM cells that are turned around at each level. Algorithms 3 to 7 do not suffer from these flaws.

Table 1: Comparison of consolidation algorithms

Algorithm	1	2	3	4	5	6	7
Complexity	**High**	**High**	Low	Medium	>Medium	>Medium	>>Medium
Transient Response	Fast	Medium	Medium	**Slow**	Fast for overload	Fast for overload	Very fast for overload
Consolidation Noise	High	Medium	**High**	Low	Low	Low	Low
BRM to FRM Ratio at Root	1	<1	≤1	≤1	**maybe>1**	$lim = 1$	$lim = 1$
Sensitivity to # branch points	High	High	Low	Medium	>Medium	Medium	Medium

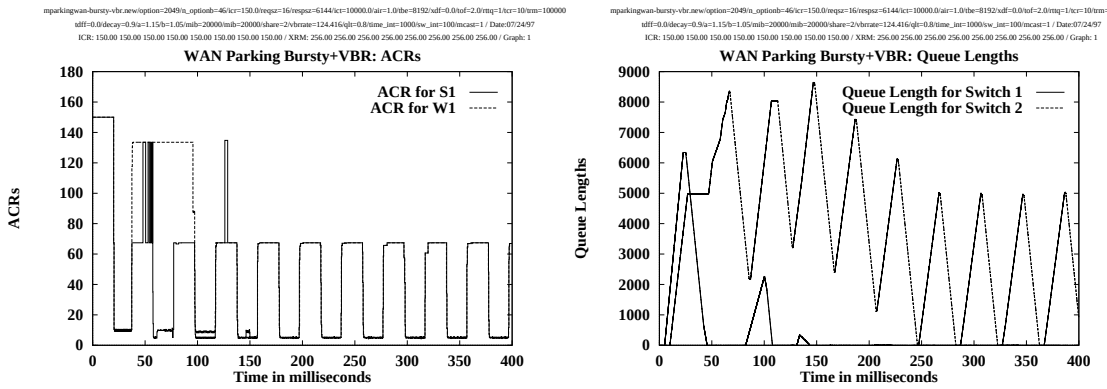
The comparison clearly indicates that algorithms 1 and 2 suffer from many problems. Algorithm 3 is good, except for the consolidation noise problems which lead to unacceptable performance as seen in the figures (especially 22 and 29). Algorithm 4 provides reasonable performance, but has a slow transient response, which is overcome by the algorithms we proposed (5, 6 and 7). Algorithm 4 and the new algorithms are slightly more complex than algorithm 3, but this can be well worth the performance benefits gained, especially with algorithm 7. *Algorithm 7 avoids congestion, while eliminating the consolidation noise problem.*

References

[1] You-Ze Cho and Myong-Yong Lee. An efficient rate-based algorithm for point-to-multipoint ABR service. Submitted to IEEE GLOBECOM 1997, April 1997.

[2] Sonia Fahmy, Raj Jain, Shivkumar Kalyanaraman, Rohit Goyal, Bobby Vandalore, Xiangrong Cai, and Seong-Cheol Kim. Performance analysis of ABR point-to-multipoint connections for bursty and non-bursty traffic with and without VBR background. ATM-FORUM/97-0422, April 1997.

[3] 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.



(a) Transmitted Cell Rate

(b) Queue Length

Figure 6: Results for WAN parking lot configuration with bursty, infinite and VBR connections [Algorithm 1]

[4] Doug Hunt. Open issues for ABR point-to-multipoint connections. ATM-FORUM/95-1034, August 1995.

[5] Raj Jain, Sonia Fahmy, Shivkumar Kalyanaraman, and Rohit Goyal. ABR switch algorithm testing: A case study with ERICA. ATM-FORUM/96-1267, October 1996.

[6] Raj Jain, Shivkumar Kalyanaraman, Rohit Goyal, Sonia Fahmy, and Ram Viswanathan. ERICA switch algorithm: A complete description. ATM-FORUM/96-1172, August 1996.

[7] Wenge Ren. Congestion control for data traffic over ATM networks. PhD proposal. Available through W. Ren's home page, 1996.

[8] Wenge Ren, Kai-Yeung Siu, and Hiroshi Suzuki. On the performance of congestion control algorithms for multicast ABR service in ATM. In *Proceedings of IEEE ATM'96 Workshop, San Francisco*, August 1996.

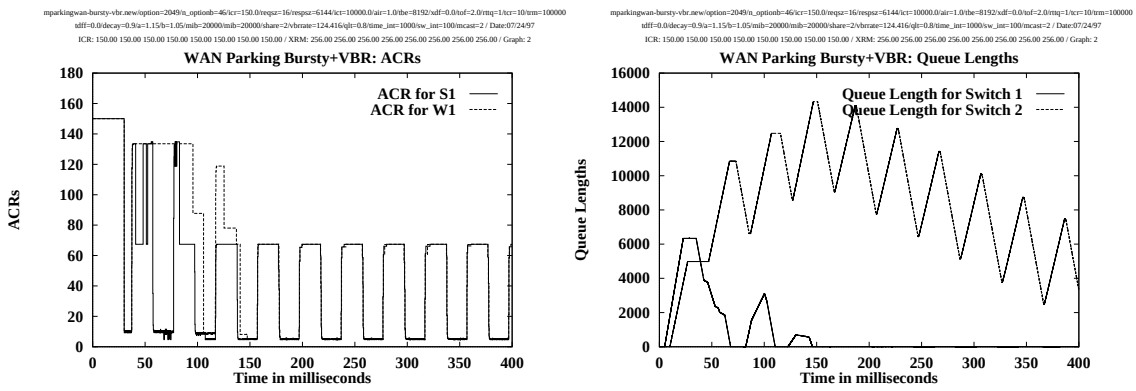
[9] Lawrence Roberts. Rate based algorithm for point to multipoint ABR service. ATM-FORUM/94-0772R1, November 1994.

[10] Lawrence Roberts. Addition to TM spec 4.0 on point-to-multipoint. ATM-FORUM/95-0339, April 1995.

[11] Lawrence Roberts. Point-to-multipoint ABR operation. ATM-FORUM/95-0834, August 1995.

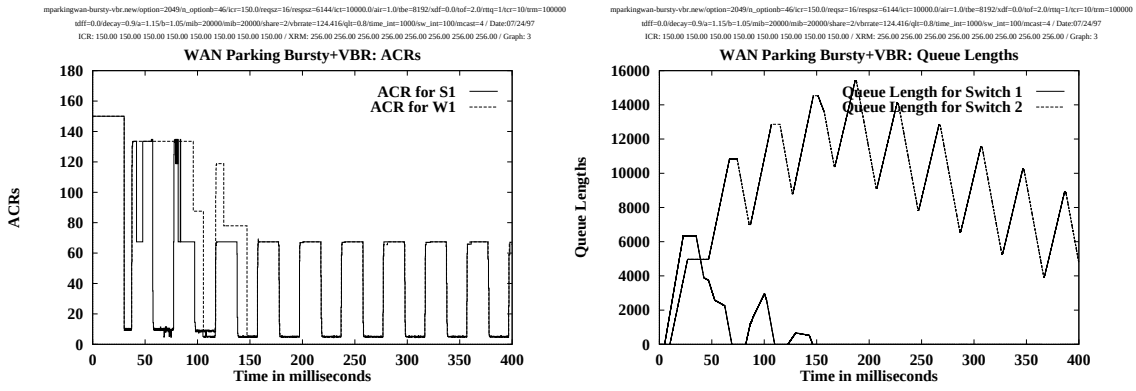
[12] Kai-Yeung Siu and Hong-Yi Tzeng. Congestion control for multicast service in ATM networks. In *Proceedings of the IEEE GLOBECOM*, volume 1, pages 310–314, 1995.

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



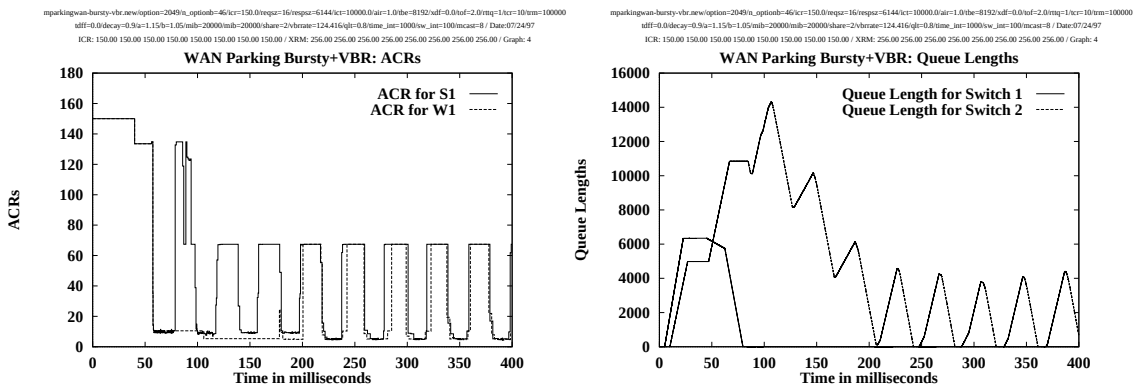
(a) Transmitted Cell Rate (b) Queue Length

Figure 7: Results for WAN parking lot configuration with bursty, infinite and VBR connections [Algorithm 2]



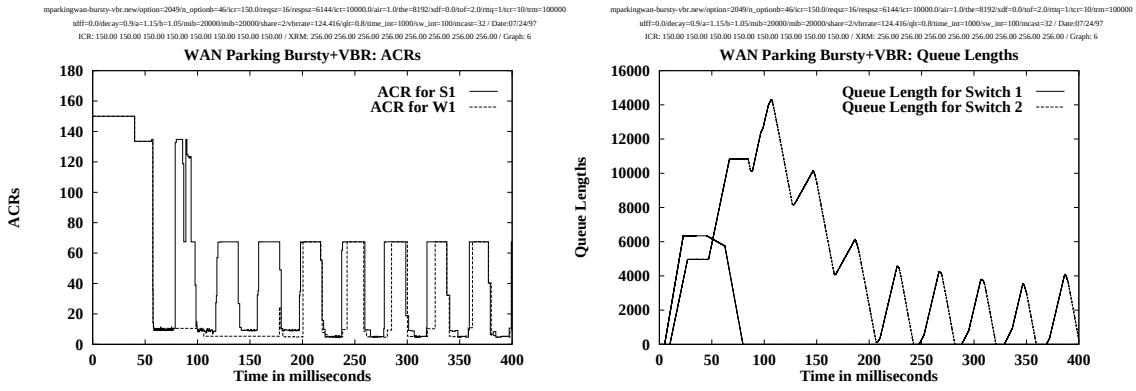
(a) Transmitted Cell Rate (b) Queue Length

Figure 8: Results for WAN parking lot configuration with bursty, infinite and VBR connections [Algorithm 3]



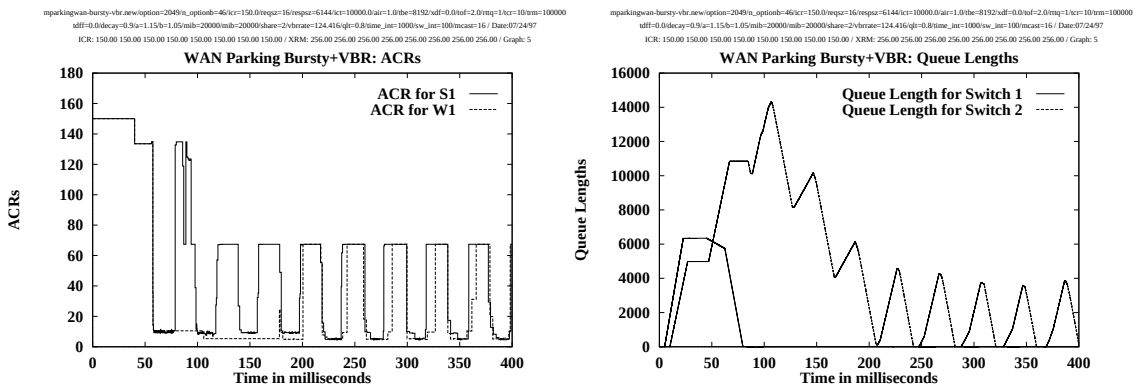
(a) Transmitted Cell Rate (b) Queue Length

Figure 9: Results for WAN parking lot configuration with bursty, infinite and VBR connections [Algorithm 4]



(a) Transmitted Cell Rate (b) Queue Length

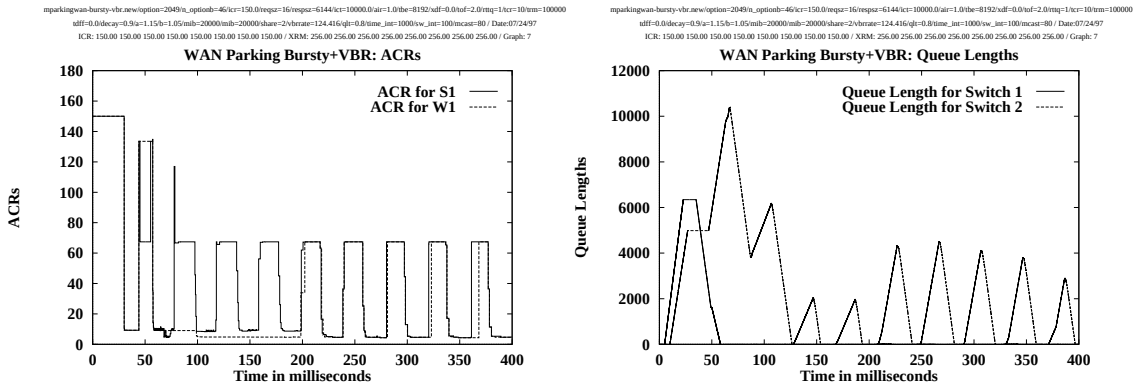
Figure 10: Results for WAN parking lot configuration with bursty, infinite and VBR connections [Algorithm 5]



(a) Transmitted Cell Rate

(b) Queue Length

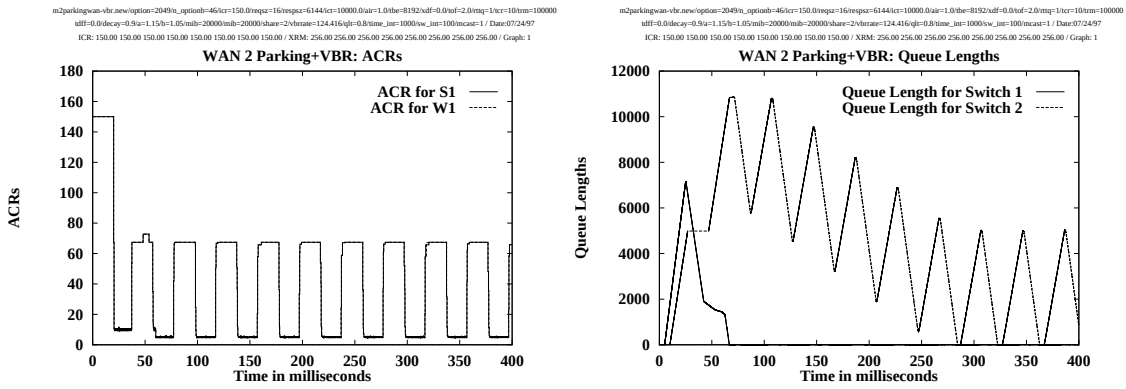
Figure 11: Results for WAN parking lot configuration with bursty, infinite and VBR connections [Algorithm 6]



(a) Transmitted Cell Rate

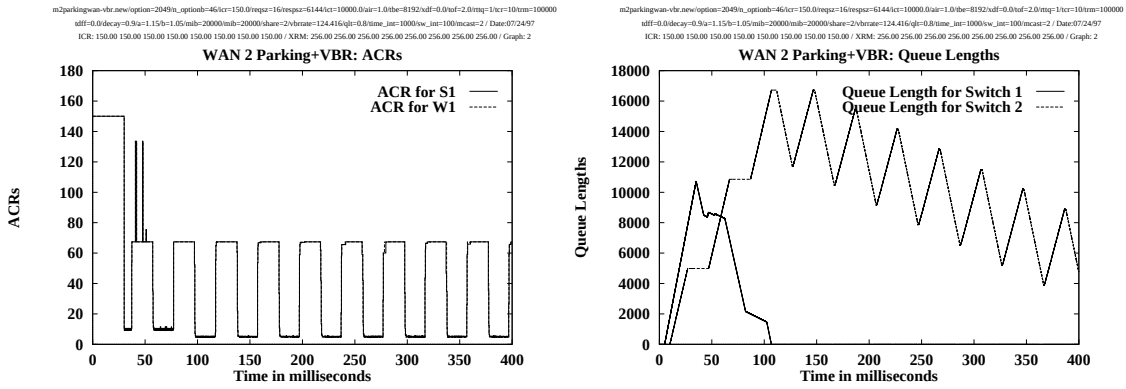
(b) Queue Length

Figure 12: Results for WAN parking lot configuration with bursty, infinite and VBR connections [Algorithm 7]



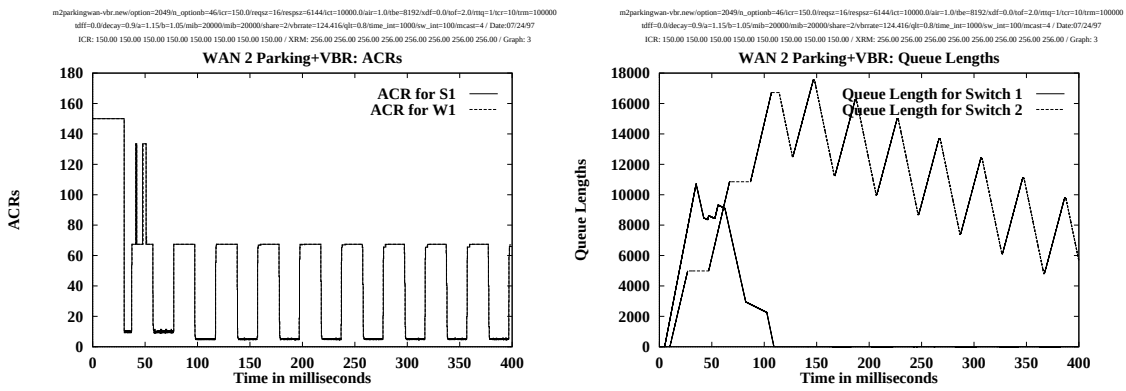
(a) Transmitted Cell Rate (b) Queue Length

Figure 13: Results for a WAN parking lot configuration with 2 infinite and one VBR connections [Algorithm 1]



(a) Transmitted Cell Rate (b) Queue Length

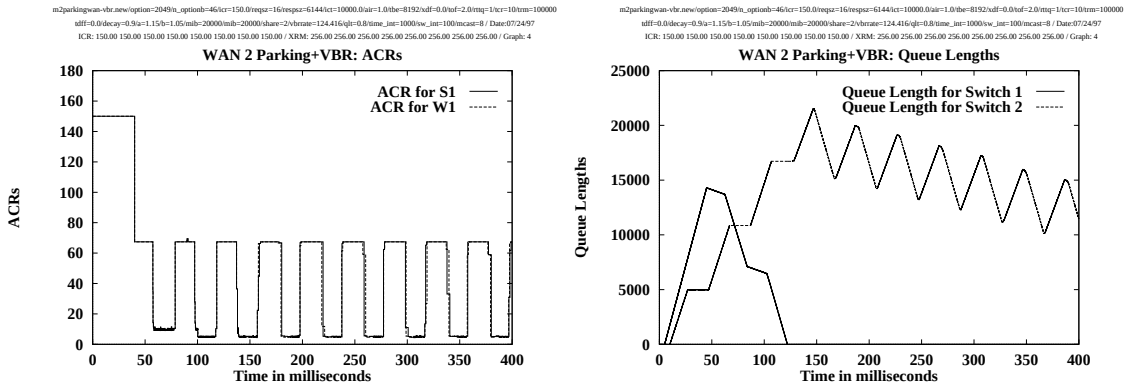
Figure 14: Results for a WAN parking lot configuration with 2 infinite and one VBR connections [Algorithm 2]



(a) Transmitted Cell Rate

(b) Queue Length

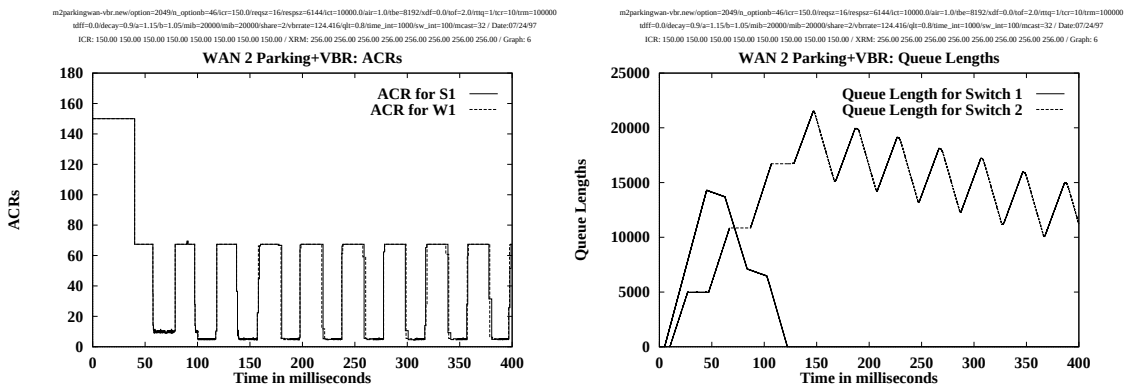
Figure 15: Results for a WAN parking lot configuration with 2 infinite and one VBR connections [Algorithm 3]



(a) Transmitted Cell Rate

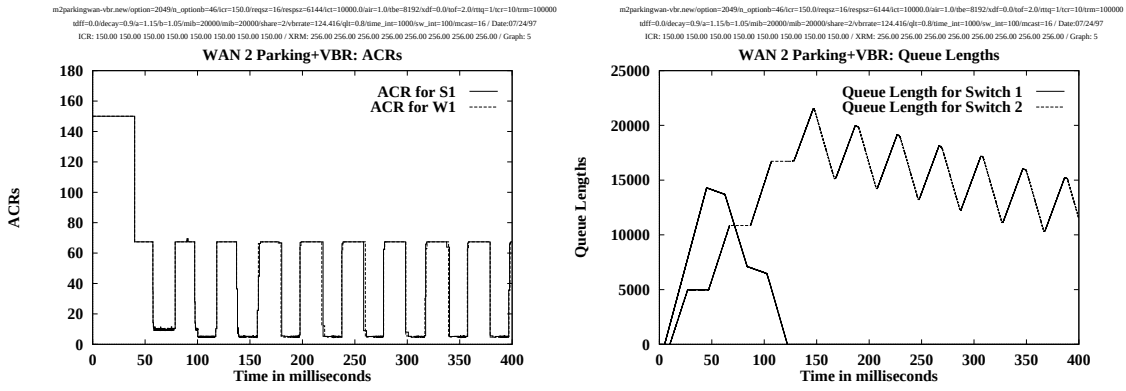
(b) Queue Length

Figure 16: Results for a WAN parking lot configuration with 2 infinite and one VBR connections [Algorithm 4]



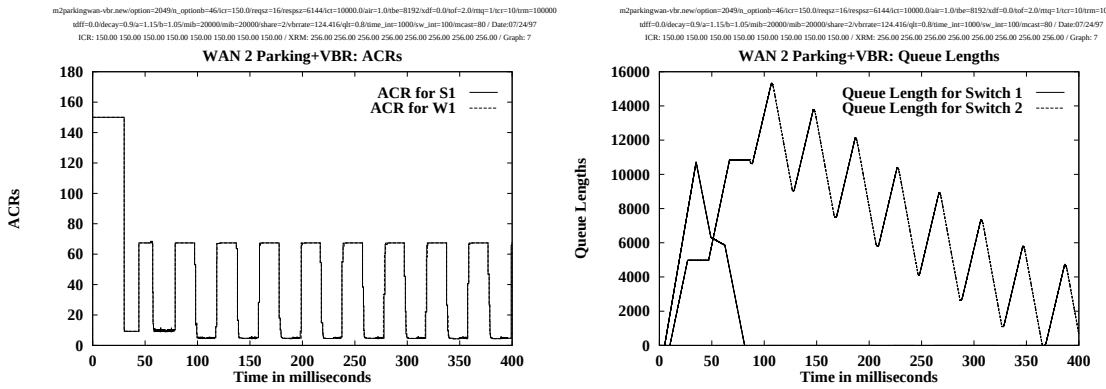
(a) Transmitted Cell Rate (b) Queue Length

Figure 17: Results for a WAN parking lot configuration with 2 infinite and one VBR connections [Algorithm 5]



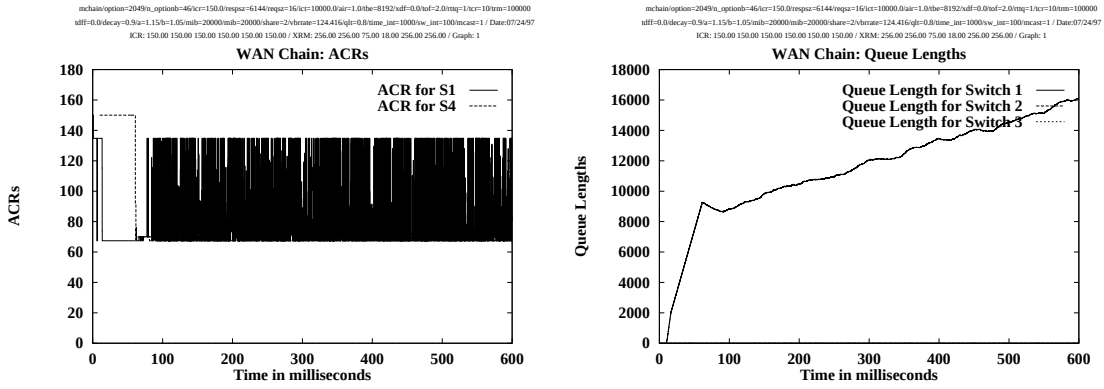
(a) Transmitted Cell Rate (b) Queue Length

Figure 18: Results for a WAN parking lot configuration with 2 infinite and one VBR connections [Algorithm 6]



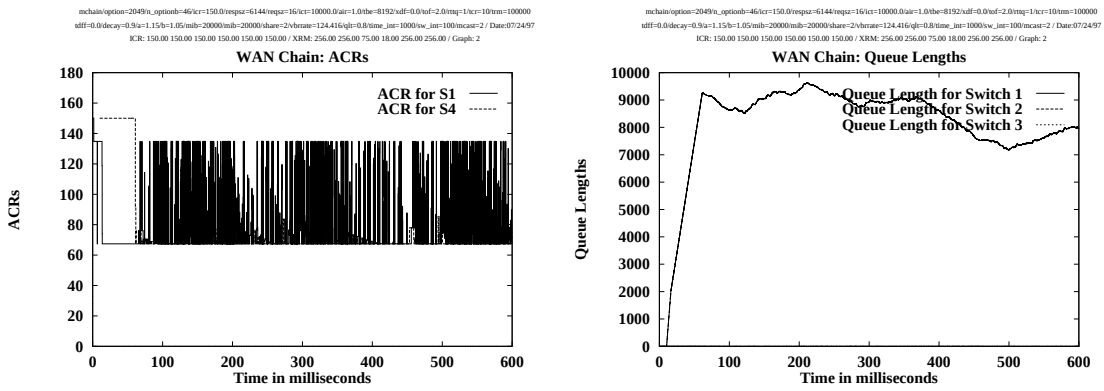
(a) Transmitted Cell Rate (b) Queue Length

Figure 19: Results for a WAN parking lot configuration with 2 infinite and one VBR connections [Algorithm 7]



(a) Transmitted Cell Rate (b) Queue Length

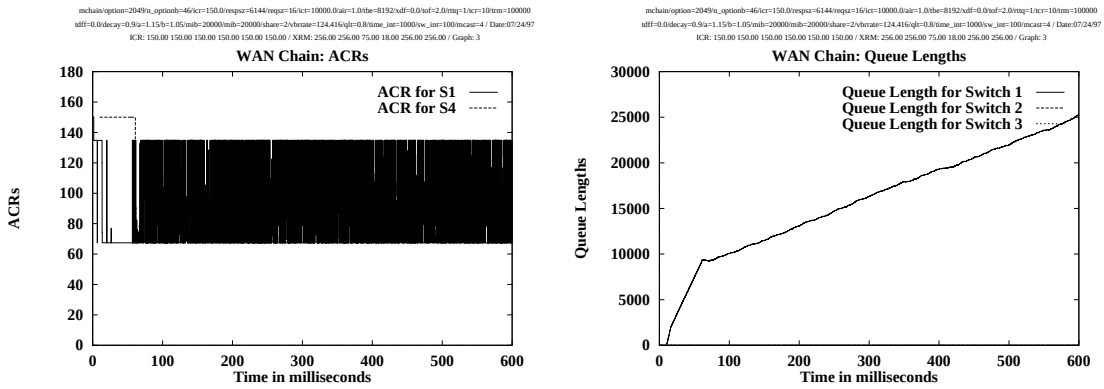
Figure 20: Results for a WAN chain configuration [Algorithm 1]



(a) Transmitted Cell Rate

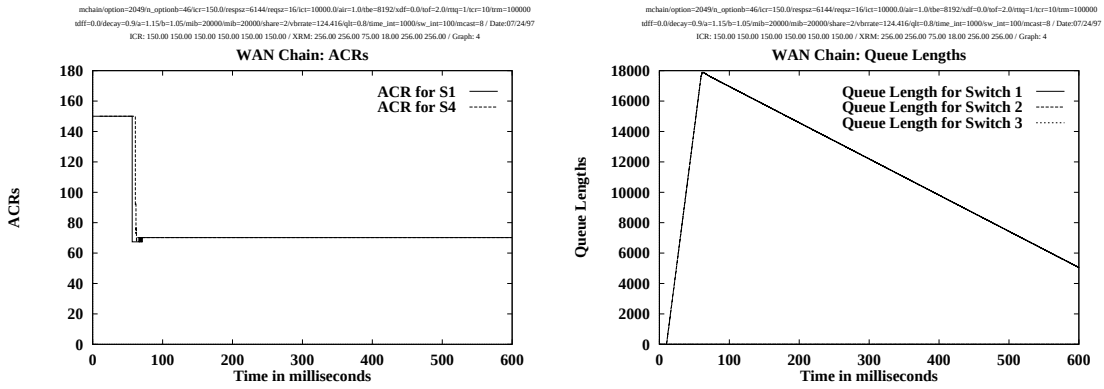
(b) Queue Length

Figure 21: Results for a WAN chain configuration [Algorithm 2]



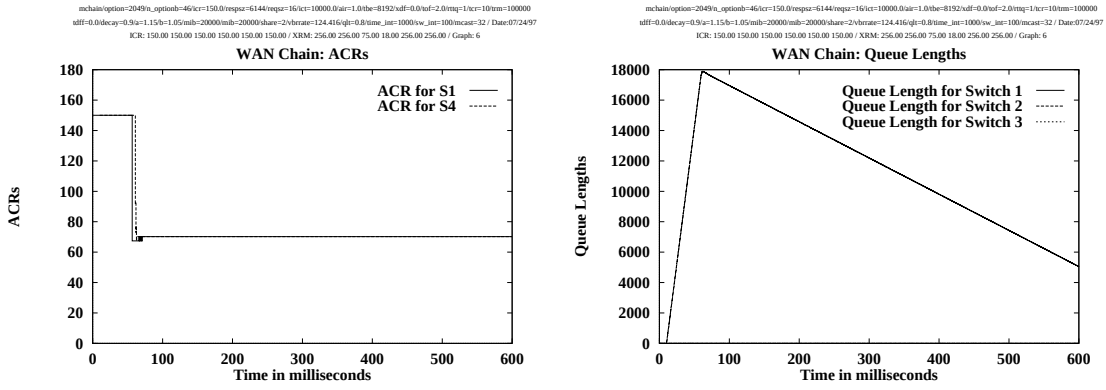
(a) Transmitted Cell Rate (b) Queue Length

Figure 22: Results for a WAN chain configuration [Algorithm 3]



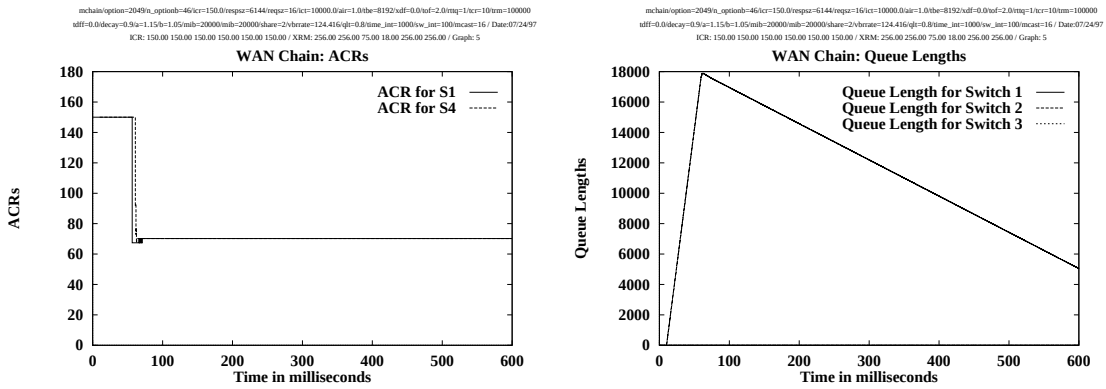
(a) Transmitted Cell Rate (b) Queue Length

Figure 23: Results for a WAN chain configuration [Algorithm 4]



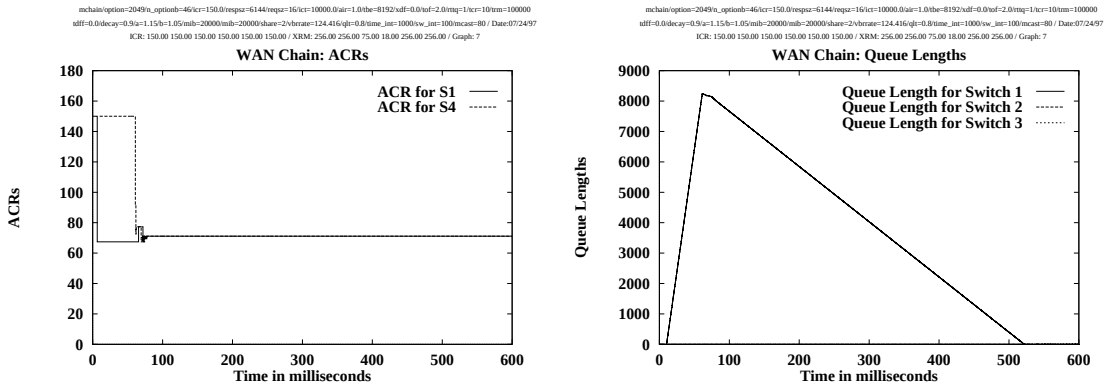
(a) Transmitted Cell Rate (b) Queue Length

Figure 24: Results for a WAN chain configuration [Algorithm 5]



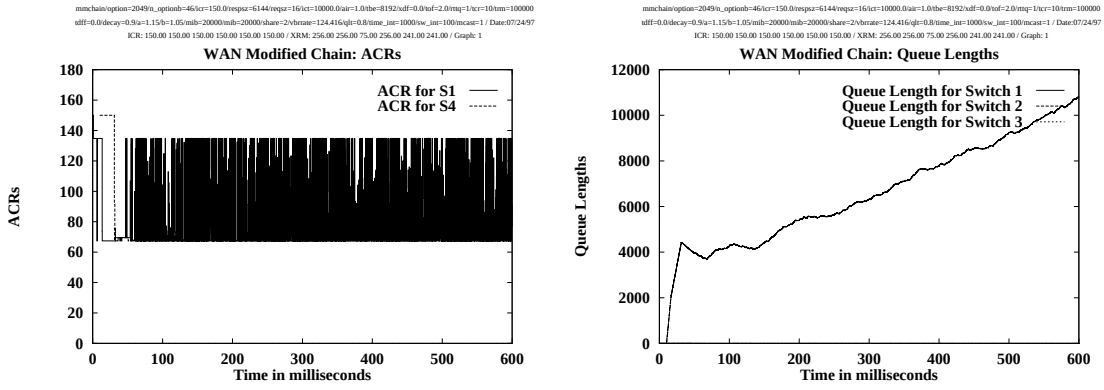
(a) Transmitted Cell Rate (b) Queue Length

Figure 25: Results for a WAN chain configuration [Algorithm 6]



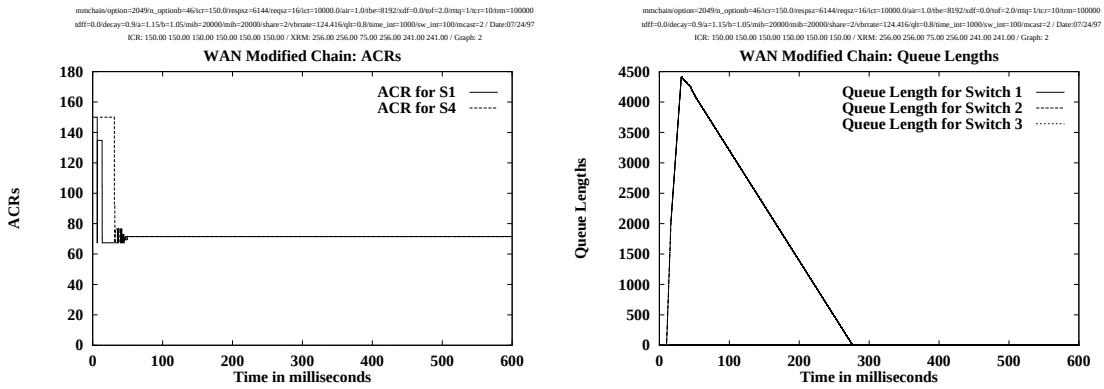
(a) Transmitted Cell Rate (b) Queue Length

Figure 26: Results for a WAN chain configuration [Algorithm 7]



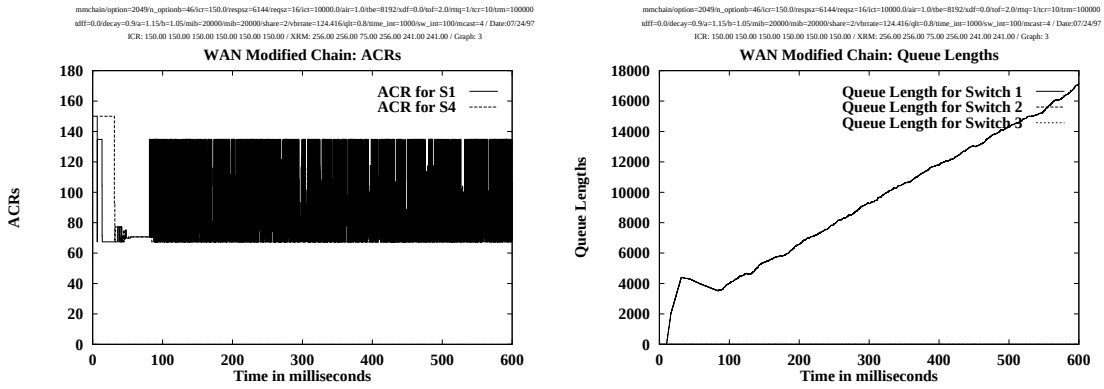
(a) Transmitted Cell Rate (b) Queue Length

Figure 27: Results for a modified chain configuration [Algorithm 1]



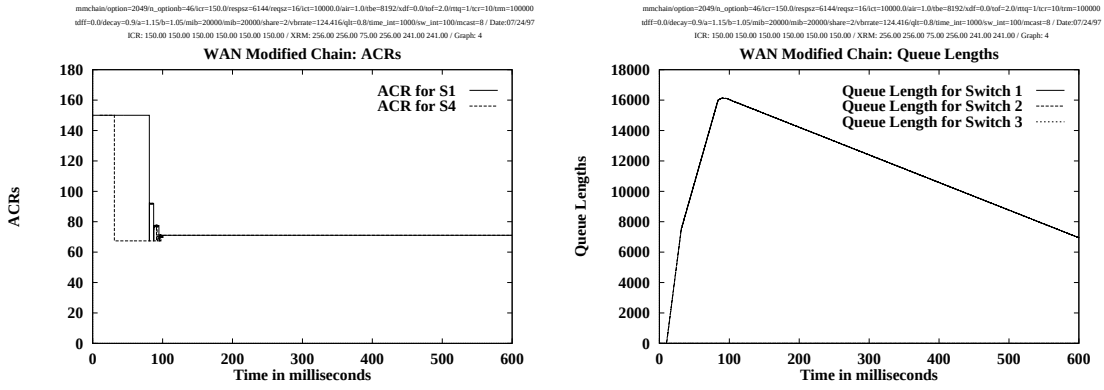
(a) Transmitted Cell Rate (b) Queue Length

Figure 28: Results for a modified chain configuration [Algorithm 2]



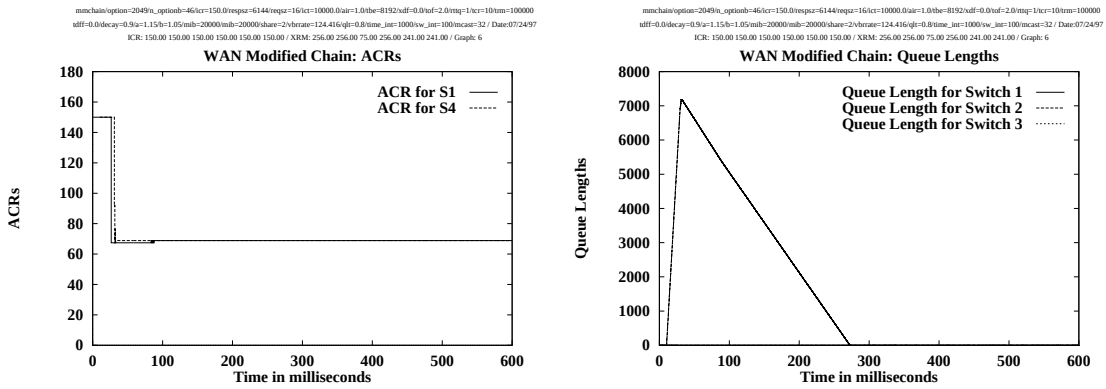
(a) Transmitted Cell Rate (b) Queue Length

Figure 29: Results for a modified chain configuration [Algorithm 3]



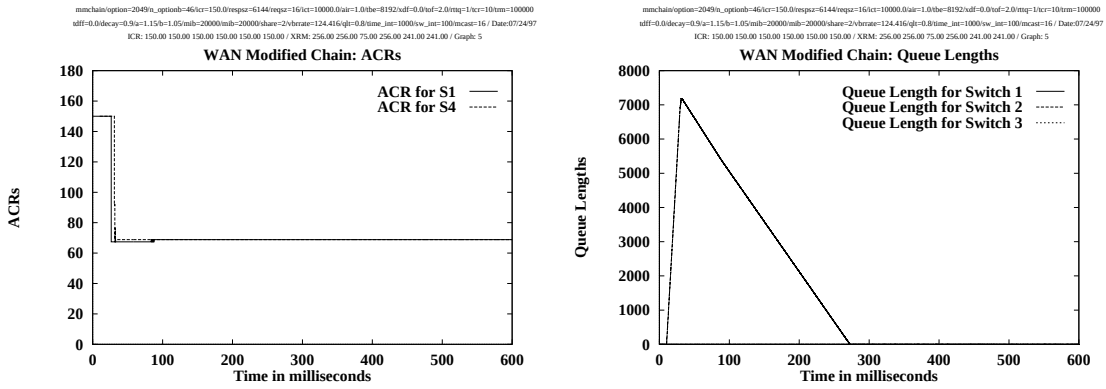
(a) Transmitted Cell Rate (b) Queue Length

Figure 30: Results for a chain modified configuration [Algorithm 4]



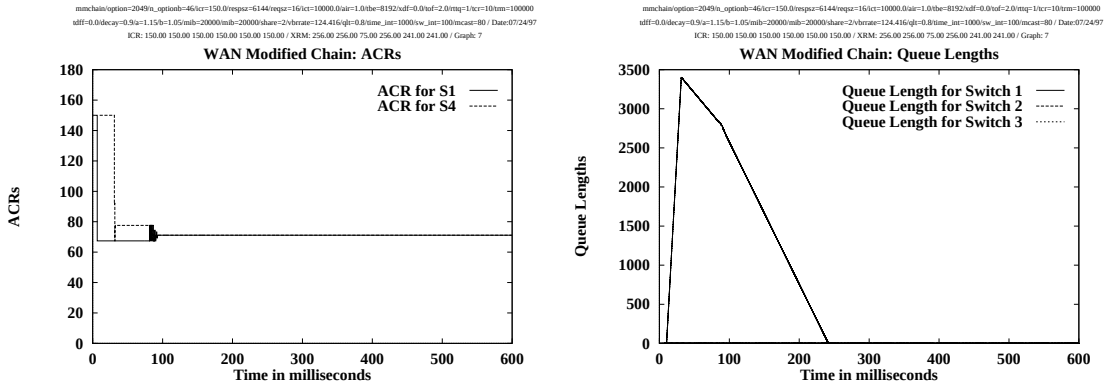
(a) Transmitted Cell Rate (b) Queue Length

Figure 31: Results for a modified chain configuration [Algorithm 5]



(a) Transmitted Cell Rate (b) Queue Length

Figure 32: Results for a modified chain configuration [Algorithm 6]



(a) Transmitted Cell Rate (b) Queue Length

Figure 33: Results for a modified chain configuration [Algorithm 7]