Evaluation of Flow Control Algorithms for ABR Multipoint Services

Uyen Trang Nguyen
Department of Computer Science, York University, Toronto, Canada, email: utn@cs.yorku.ca

Abstract

In this paper, we first present a simulation-based evaluation of existing multipoint-to-point (mp-p) flow control schemes for ABR (Available Bit Rate) services in ATM (Asynchronous Transfer Mode) networks. The evaluation shows that these schemes suffer from one or more of the following problems: non-compliance with the selected bandwidth allocation definition (BAD), link under-utilization, and prolonged rate fluctuations. Moreover, most of them implement only one BAD, namely source-based allocation. We then propose a mp-p flow control algorithm, the WSB (weighted-source-based) algorithm, that is effective and supports a large set of BADs. Experimental results are presented to demonstrate the advantages of the WSB algorithm: correct rate allocations with fast convergence, maximum link utilization, and minimal rate oscillations.

1. Introduction

Multipoint (mp) communications support the exchange of information among a set of participants in an efficient manner. Multipoint communications encompass an important class of applications whose examples are tele-metering, synchronization of replicated databases, audio/video conferencing, and distance education.

ABR is a best-effort service with no bandwidth guarantee, except a minimum cell rate. The bandwidth left over after CBR (Constant Bit Rate) and VBR (Variable Bit Rate) usage is divided evenly among ABR sources. ABR traffic is regulated using explicit rate (ER) control [1].

Multipoint-to-multipoint flow control can be implemented by combining point-to-multipoint and mp-p control algorithms [5, 12, 14]. Although there exist many mp-p flow control schemes [6, 9, 11, 12, 13], several issues remain to be answered: bandwidth allocation definitions (BADs), and the effectiveness and performance of those schemes. BADs refer to criteria for allocating ABR bandwidth among multipoint connections, or between unicast (point-to-point) and multipoint connections. Several BADs have been suggested for multipoint communications [4, 9, 11]. However, researchers have not yet agreed on which definitions are most suitable for ABR multipoint applications. Thus a mp-p flow control scheme should support several BADs so that switch vendors can configure their switches quickly and easily according to clients' requirements.

A mp-p flow control scheme is *effective* if it allocates ABR bandwidth to sources according to the selected BAD and maximizes the utilization of ABR capacity. Desirable performance measures of a multipoint flow control scheme are fast convergence and minimal rate oscillations.

In this paper, we first present a simulation-based evaluation of existing mp-p flow control schemes. The evaluation shows that these schemes suffer from one or more of the following problems: non-compliance with the selected BAD, link under-utilization, and prolonged rate fluctuations. Moreover, most of them implement only source-based allocation. This inflexibility can limit the application of those schemes. We then propose a mp-p flow control algorithm, the WSB algorithm, that is effective and supports a large set of BADs. Experimental results are presented to demonstrate the advantages of the WSB algorithm: correct rate allocations with fast convergence, maximum link utilization, and minimal rate oscillations.

The paper is organized as follows. Section 2 summarizes related work on BADs and ABR mp-p flow control. Section 3 presents an evaluation of the rate calculation algorithms implemented in existing mp-p flow control schemes. The proposed WSB algorithm is described in section 4, followed by simulation results. Section 5 concludes the paper.

2. Related Work

For unicast connections, the most commonly used BAD is max-min "fairness" [7]. The BAD becomes more complicated when mp connections are involved since such a mp VC is shared by several sources. The suggested BADs based on sources are:

Characteristic	**RSS**	**MC**	**NK**	**Pao's**	**FJ**	**WSB**
Base unicast RCA	(any)	ERICA [8]	FMMRA [3]	ERICA	ERICA	ERICA
Multipoint RCA	n/a	using VC weights			using source CCRs	
Source-based	x	x	x	x	x	x
VC/source-based		x	x			x
Weighted-source-based		x	x			x

Table 1: Characteristics of the existing mp-p flow control algorithms

– **Source-based** [4]: divides ABR bandwidth on a link evenly among active *sources* sharing the link, ignoring whether a source belongs to a unicast or a mp VC.
– **VC/source-based** [4]: first allocates the ABR bandwidth evenly among the VCs, unicast or multipoint, and then divides the allocated bandwidth of each VC evenly among the active *sources* in the VC.
– **Weighted-source-based** [9]: requires every source to be assigned a weight. The weight of any unicast source is 1. The weight of a source of a mp connection i is w_i, where $0 < w_i \leq 1$. For two different mp connections i and j, we can have $w_i \neq w_j$. ABR bandwidth is allocated to the sources proportionally to their weights. Note that when $w_i = 1$, we have source-based allocation. Weighted-source-based definition provides a compromise between the above two extreme definitions.

There are also BADs based on flows [4, 11]. However, these BADs suffer from the beat-down problem: the farther away a source from the destination in terms of the number of merge points, the smaller the rate it is allocated [4]. We thus do not consider BADs based on flows in this paper.

Ren, Siu, and Suzuki propose a mp-p flow control scheme [13], the RSS algorithm[1], that supports only source-based BAD. The authors did not provide a specific multipoint rate calculation algorithm (RCA), but claimed that any unicast RCA can be applied directly to their mp-p scheme without modifications. However, if data forwarding uses VC merge [4] and the unicast RCA uses per-source accounting (e.g., the ERICA algorithm [8]), the mp-p algorithm will not allocate ABR bandwidth correctly, as previously demonstrated by simulation results [9]. Ren *et al.* also suggest a framework for mp-mp flow control [14] that uses the RSS algorithm. The resulting mp-mp scheme is thus not effective.

Cavendish and Gerla also propose a mp-mp flow control scheme [2], but do not address the mp-p case. A mp-mp connection with N members is implemented as N p-mp connections, each having one sender and $N-1$ receivers. This scheme is thus not scalable as N grows.

Pao proposes a mp-mp flow control scheme [12] that combines the p-mp flow control algorithm designed by Fahmy *et al.* [5] and a mp-p algorithm. The proposed mp-p RCA can be considered as the main contribution of Pao's paper, which will be evaluated in Section 3.

Other existing mp-p flow control schemes are the FJ algorithm by Fahmy, Jain, *et al.* [6], the MC algorithm by Moh and Chen [9], and the NK algorithm by Nguyen and Katzela [11].

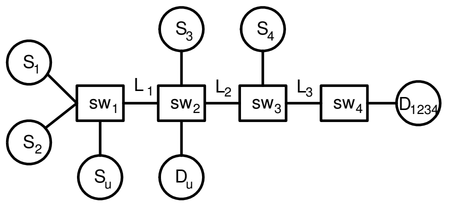
Table 1 summarizes the characteristics of the above mp-p flow control schemes (excluding the scheme by Cavendish and Gerla as it does not address the mp-p case). The multipoint RCAs of these schemes are discussed and analyzed in the next section.

[1]Named using the first letters of the authors' last names. We will use this naming convention throughout the paper to denote different mp-p flow control schemes.

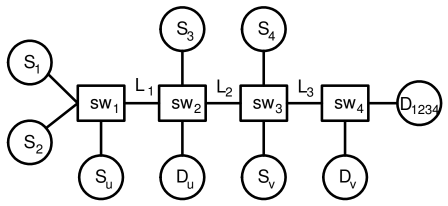
3. Evaluation of Multipoint RCAs

To assess the effectiveness and performance of the multipoint RCAs, we have used simulation, a common method for evaluating ATM protocols. We started with the ATM network simulator version 4.1 developed by the American National Institute of Standards and Technology [10], and added code to implement the MC, FJ, NK and Pao's algorithms. We did not include the RSS algorithm in the evaluation because it does not provide a specific RCA. The performance of a RCA is evaluated based on its convergence time, and rate oscillations during the transient period as well as after convergence. We use the ERICA algorithm as the base RCA because it is well designed and documented, although any effective unicast RCA can be used. The same base RCA is used in all the mp-p flow control schemes to be evaluated in order to make the comparison fair.

We use the configurations depicted in Figure 1 for the evaluation. In Configuration 1, mp-p sources S_1, S_2, S_3 and S_4 all send data to destination D_{1234}, while unicast source S_u sends to destination D_u. Configuration 2 is the same as Configuration 1, except that a unicast



(a) Configuration 1



(b) Configuration 2

Figure 1: Configurations used for experiments

Configuration	**Multipoint source weight**	S_u	S_1	S_2	S_3	S_4	S_v
1 in Figure 1(a)	1.00 (source-based)	16.67	16.67	16.67	58.33	58.33	
	0.50 (VC/source-based)	25.00	12.50	12.50	62.50	62.50	
	0.75	20.00	15.00	15.00	60.00	60.00	
2 in Figure 1(b)	1.00 (source-based)	16.67	16.67	16.67	38.88	38.88	38.88
	0.50 (VC/source-based)	25.00	12.50	12.50	31.25	31.25	62.50
	0.75	20.00	15.00	15.00	36.00	36.00	48.00

Table 2: Sources' rates in Mbps

connection with source S_v and destination D_v is added. All the links have a capacity of 150 Mbps, except link L_1 whose capacity is 50 Mbps. The distance between switches is 1000 km. Terminal-switch distance is 1km. VC merge is simulated, and the packet size is 512 bytes. The sources are persistent, and start sending data at the same time with the same initial cell rate ICR = 25 Mbps, unless otherwise stated. The duration of each experiment is 1000 msec. Table 2 lists the sources' rates for both configurations using different BADs. In every experiment, we recorded the ACRs of the sources and the utilization of links L_2 and L_3 (link L_1 is 100% utilized in all the experiments).

At a merge point of a mp-p VC (e.g., switch sw_3), the outgoing link carrying data of all the branches is called the *merged link* (e.g., link L_3). A branch/VC that cannot use up the bandwidth the switch offers due to a bottleneck elsewhere is called a *bottlenecked* branch/VC. Otherwise, it is called an *unconstrained* branch/VC.

RCAs in mp-p flow control schemes can be divided broadly into two groups: one uses VC weights, and the other uses source CCRs (current cell rates).

3.1. RCAs Using VC Weights

The weight of any unicast VC is 1. The weight W_k of a mp VC k on a link L is $W_k = N_k \times w_k$, where w_k is the weight of a source of VC k, and N_k is the number of sources of VC k merging on link L. The ERICA algorithm is modified so that each VC is offered an amount of bandwidth proportional to its weight [9, 11, 12]. If the current switch is a merge point of the mp VC, the ER allocated to the mp VC is then divided among the branches merging at the switch. The division can be done using branch weights [9, 11], aggregate CCRs of the branches, or a combination of both [12].

Rate Division Using Branch Weights. The MC and NK algorithms employ this method. The weight of a branch j of a mp VC k is defined as $w_{j,k} = n_{j,k} \times w_k$, where $n_{j,k}$ is the number of sources merging on this branch, and w_k is the weight of sources of VC k. Each branch is allocated an amount of bandwidth proportional to its weights: $branch\_ER_j = VC\_ER_k \times w_{j,k}/W_k$. If one of the branches at the merge point encounters a bottleneck upstream, one of the following problems may result: (1) non-compliance with BAD, and (2) link underutilization.

The first problem occurs when the merged link is shared by the mp VC and one or more unconstrained VCs. In Figure 1(b), link L_3 is shared by the mp VC and the unconstrained unicast VC (S_v, D_v). Assume source-based allocation. The ACR graph in Figure 4(a) shows that, the ARCs of sources S_3, S_4 and S_v are 30, 30 and 56.67 Mbps respectively, while they each should have got an equal rate of 38.88 Mbps according to source-based BAD (see Table 2). Sources S_3 and S_4 cannot use up their shares of 38.88 Mbps because switch sw_3 does not know about the upstream bottleneck (link L_1) in order to redistribute the unused bandwidth from S_1 and S_2 to S_3 and S_4. The design of the RCA then allows S_v to raise its rate in order to obtain an optimal load factor of 1 on link L_3. S_v thus acquires a bigger share than S_3 and S_4 on L_3, violating source-based BAD.

When a merged link carries only multipoint VCs, each of which has at least one bottlenecked branch (e.g., link L_3 in Configuration 1, and the branch going to switch sw_2), the link may be under-utilized, even though the ERs of the unconstrained branches (e.g., the branch going to source S_4) could be increased for higher utilization. The following experiment, which uses Configuration 1 and source-based BAD, illustrates this problem. The ACR graph in Figure 4(b) indicates that S_3 and S_4 transmit at 37.5 Mbps, while their ACRs would have been 58.33 Mbps according to source-based allocation (see Table 2). As a result, the utilization of link L_3 varies between 67% and 79% instead of being 100% had S_3 and S_4 been able to claim the unused bandwidth of S_1 and S_2. Again, switches sw_2 and sw_3 are not aware of the upstream bottleneck (link L_1) in order to re-allocate the unused bandwidth from S_1 and S_2 to S_3 and S_4.

The RCA by Pao [12] attempts to alleviate the problem of link under-utilization by dividing the bandwidth at merge points using a combination of weights and aggregate CCRs of the branches.

Rate Division Using Weights and Aggregate CCRs of Branches. The aggregate CCR of a branch is the sum of the CCRs of the sources (belonging to the mp VC) whose traffic merging on this branch. The ER of a branch is computed using two ER values: one proportional to the weight of the branch, and the other, to the aggregate CCR of the branch, as follows: $branch\_ER_j = VC\_ER_k \times (w_{j,k}/W_k + branch\_CCR_{j,k}/VC\_CCR_k)/2$.

Our simulation results show that using both branch weights and aggregate CCRs indeed helps improve link utilization. However, it still does not ensure the maximum possible link utilization due to the partial use of branch weights for dividing the VC ER. This is illustrated by the following experiment that was run on Configuration 1 with source-based allocation. The ACR graph in Figure 4(c) shows that the ACRs of S_3 and S_4 vary around 49 Mbps and 46 Mbps respectively, while they each should have been 58.33 Mbps. Consequently, the utilization of link L_3 concentrates in the range of 80% - 92%, instead of being 100%.

(a)
Upon receiving an FRM cell from branch j of VC k:

```
/*update $received\_FRM[k][j]$ and max source CCR*/
$received\_FRM[k][j]$ = true;
if ($first\_FRM\_in\_this\_interval[k]$ == true) {
   $ccr[k] = $ FRM.CCR;
   $first\_FRM\_in\_this\_interval[k] = $ false;
}
else $ccr[k] = \max(ccr[k],$ FRM.CCR);
Forward the FRM cell downstream to next switch;
```

(b)
Upon receiving a BRM cell from VC k:

```
/* calculate source ER */
1: if ($load\_factor > 1 + \delta$)
2:     $er\_calculated = ccr[k]/load\_factor$;
3: else $er\_calculated$ = max ($ccr[k]/load\_factor$,
4:          $max\_ER\_previous\_interval$);
5: $er\_calculated$=min($er\_calculated$,$ABR\_capacity$);
6: $max\_ER\_current\_interval$ =
7:    max($max\_ER\_current\_interval$, $er\_calculated$);
8: $er\_calculated$ = min (BRM.ER, $er\_calculated$);
9: Generate_BRMs($k$, $er\_calculated$);
```

(c)

```
Generate_BRMs($k$, $er\_calculated$) {
for every branch $j$ of VC $k$
  if ($received\_FRM[k][j]$) {
    Generate a BRM cell;
    Copy the fields from the original BRM cell;
    BRM$_j$.ER = $er\_calculated$;
    Send the new BRM cell upstream along branch $j$;
  }
}
```

(d)
At the end of every measurement interval:

```
Recalculate $ABR\_target\_capacity$ and $input\_rate$
(refer to reference [8]);
$load\_factor = input\_rate/ABR\_target\_capacity$;
$max\_ER\_previous\_interval =$
        $(1-\alpha) \times max\_ER\_current\_interval$
        $+\alpha \times max\_ER\_previous\_interval$;
$max\_ER\_current\_interval = 0$;
for every VC $i$
  $first\_FRM\_in\_this\_interval[i] = $ true;
/* Note: recommended value of $\delta$ is 0.1;
recommended value of $\alpha$ is 0.1 [6] */
```

Figure 2: The FJ rate calculation algorithm [6]

An interesting question to be explored is whether the use of aggregate CCRs of branches alone: $branch\_ER_j = VC\_ER_k \times branch\_CCR_{j,k}/VC\_CCR_k$ would be an effective method for dividing VC ERs at merge points. We implemented this method, and simulated source-based allocation on Configuration 1. The results given in Figure 4(d) show that link L_3 is indeed fully utilized, at 100%. Nonetheless, the rate allocation is no longer source-based correct: S_u gets a higher rate than S_1 and S_2 on link L_1. Note also that source's rates continue to fluctuate after the transient period.

3.2. RCA Using Source CCRs

The FJ algorithm uses this method, and supports only source-based BAD [6]. The algorithm is outlined in Figure 2. For every output link L, the switch maintains the maximum source CCR of every VC (carried in FRM cells), and the maximum ER allocated in the previous interval, $max\_ER\_previous\_interval$. Depending on the current value of the load factor on link L, every source must scale up or down its CCR by the load factor in order to maintain an optimal load of 1 on the link (Figure 2(b), lines 1−3). In addition, if the load factor is less than $1 + \delta$ (underload), a source can raise its rate to $max\_ER\_previous\_interval$ (lines 3−4) to be able to catch up with the source having the highest rate. The value of $max\_ER\_previous\_interval$ thus serves as a "reference point" for sources to aim at.

Simulation results show that this approach allows for maximum link utilization, and correct source-based allocation. One such set of simulation results is depicted by the graphs in Figure 5(a), which was obtained from running the FJ algorithm on Configuration 1. We can see that the source ACRs are calculated correctly according to source-based BAD; the utilization of link L_3 is 100%.

However, the above graphs reveal one drawback of the FJ algorithm: there is too much rate oscillation during the transient period, which results in slow convergence. Also, the application of the FJ algorithm may be limited as it supports only source-based BAD. We propose an effective RCA, the WSB algorithm, that supports weighted-source based BAD, which as a result covers both source-based and VC/source-based allocations, depending how we assign weights to multipoint sources.

4. The WSB Algorithm

Following are the reasons for much rate oscillation in the FJ algorithm. First, the ER of a VC is calculated several times while the load factor is calculated only once *per measurement interval*. If the load factor is not exactly 1, the ER will keep increasing or decreasing during that interval, causing unnecessary oscillations in the subsequent interval. Second, an FRM cell of a lower-rate source arrives at a switch as the first FRM cell of the current interval (Figure 2(a)). This causes the ERs of higher-rate sources of the mp VC to be calculated based on the low CCR value, and thus affects the load factor.

(a)
Upon receiving an FRM from branch j of VC k:

```
1: $received\_FRM[k][j]$ = true;
2: if ($first\_FRM\_in\_this\_interval[k]$ == true)
3:    if ($FRM\_count[k] < max\_count[k]$) {
4:       $temp\_ccr[k] = \max(temp\_ccr[k]$, FRM.CCR);
5:       $FRM\_count[k] = FRM\_count[k] + 1$;
6:    } else {
7:       $ccr[k] = \max(temp\_ccr[k]$, FRM.CCR);
8:       $first\_FRM\_in\_this\_interval[k]$ = false;
9:    }
10: else $ccr[k] = \max(ccr[k]$, FRM.CCR);
11: Forward the FRM cell downstream to next switch;
```

(b)
Upon receiving a BRM cell from VC k:

```
1: if ($seen\_BRM\_in\_this\_interval[k]$)
2:     $er\_calculated = last\_calculated\_ER[k]$;
3: else {
4:   if ($load\_factor > 1 + \delta$)
5:       $er\_calculated = ccr[k]/load\_factor$;
6:   else $er\_calculated$ = max ($ccr[k]/load\_factor$,
7:     $max\_ER\_previous\_interval$*$source\_weight[k]$);
8:   $er\_calculated$=min($er\_calculated$,$ABR\_capacity$);
9:   $max\_ER\_current\_interval$ =
10:       max ($max\_ER\_current\_interval$,
11:       $er\_calculated/source\_weight[k]$);
12:    /*Only one feedback per measurement interval*/
13:   $last\_calculated\_ER[k] = er\_calculated$;
14:   $seen\_BRM\_in\_this\_interval[k]$ = true;
15: }
16: $er\_calculated$ = min (BRM.ER, $er\_calculated$);
17: Generate_BRMs($k$, $er\_calculated$);
/* Note: $seen\_BRM\_in\_this\_interval[k]$ is reset to false and $FRM\_count[k]$ is reset to 0 at the end of every measurement interval (Figure 2(d)).*/
```

Figure 3: The WSB rate calculation algorithm

Following are the solutions to the above problems. First, a switch computes *only one* ER feedback value per VC per measurement interval (Figure 3(b), lines $1-3$). Second, a switch collects a few FRM cells at the beginning of a new interval, before setting variable $ccr[k]$ (Figure 3(a), lines $3-5$). This helps minimize the chances of assigning a lower CCR value to $ccr[k]$ and subsequently using it for ER calculation, because higher-rate sources generate FRM cells more frequently than lower-rate sources [1]. In our experiments, $max\_count[k] = N_k$, the number of sources of VC k merging on the link. If $N_k \geq 5$, $max\_count[k] = 5$.

We employed the idea of a "reference point" to support multipoint sources with weights, i.e., to support different BADs. Since sources of different multipoint VCs carry different weights, their ERs should not be used as $max\_ER\_previous\_interval$, the "reference point". Instead, ERs of unicast VCs are considered, because all unicast sources have the same weight of 1. Switch variables $max\_ER\_previous\_interval$ and $max\_ER\_current\_interval$ of every output port are maintained with respect to unicast sources. When the ER of a mp source is calculated, variable $max\_ER\_previous\_interval$ is scaled by the weight of the source recorded in variable $source\_weight[k]$, as shown in Figure 3(b), lines 7 and 11.

We conducted several experiments on mp-p groups of 3-15 sources to evaluate the effectiveness and performance of the WSB algorithm. Figures 5(b), (c) and (d) show the results of a set of experiments performed on Configurations 1 and 2. Note that in Figures 5(d) the ICR of the sources is set to 120 Mbps to test how well the algorithm adapts to high traffic demands. The ACR graphs show that the sources' rates are allocated properly according to each BAD as listed in Table 2; the convergence is fast in all cases, under moderate or heavy traffic, and there are no noticeable rate oscillations after convergence. The utilization of L_3 is 100% in all cases.

5. Conclusion

Multipoint-to-point flow control is essential to implement an effective mp-mp algorithm. We conducted an evaluation of existing ABR mp-p flow control schemes. Experiments show that RCAs based on aggregate data such as VC/branch weights and/or VC/branch CCRs suffer from the problems of non-compliance with the selected BAD or link under-utilization in many cases. We propose an effective RCA based on source CCRs, the WSB algorithm, that allows for maximum link utilization and fast convergence with minimal rate oscillations, as confirmed by simulation results. It is easy to extend the WSB algorithm into a mp-mp flow control scheme by combining it with an effective p-mp algorithm [5].

References

[1] ATM Forum, "ATM Forum Traffic Management Specification Version 4.0," April 1996.

[2] D. Cavendish and M. Gerla, "Rate Based Congestion Control for Many-to-Many Multicast ABR Traffic," *Proc. of SPIE Conf. on Performance and Control of Network Systems II*, Nov. 1998, pp. 122-130.

[3] F.M. Chiussi, *et al.*, "Explicit Rate ABR Schemes Using Traffic Load as Congestion Indicator," *Proc. of 6th International Conf. on Computer Communications and Networks*, Sept. 1997, pp. 76-84.

[4] S. Fahmy, *et al.*, "Fairness for ABR multipoint-to-point connections," *Proc. of SPIE Conf. on Performance and Control of Network Systems II*, Nov. 1998, pp. 131-142.

[5] S. Fahmy, *et al.*, "Design and Evaluation of Feedback Consolidation for ABR Point-to-Multipoint Connections in ATM Networks," *Computer Communications*, vol. 22, no. 12, July 1999, pp. 1085-1103.

[6] S. Fahmy, *et al.*, "Fair Flow Control for ATM-ABR Multipoint Connections," *Computer Communications*, vol. 25, no. 8, May 2000, pp. 741-755.

[7] J. M. Jaffe, "Bottleneck Flow Control," *IEEE Transactions on Communications*, vol. 29, no. 7, July 1981, pp. 954-962.

[8] S. Kalyanaraman, *et al.*, "The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 1, Feb. 2000, pp. 87-98.

[9] W.M. Moh and Y. Chen, "Design and Evaluation of Multipoint-to-Point Multicast Flow Control," *Proc. of SPIE Conf. on Performance and Control of Network Systems II*, Nov. 1998, pp. 143-154.

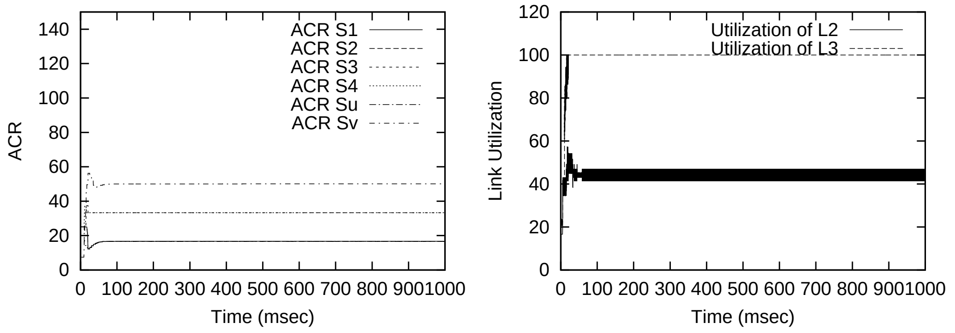
[10] The NIST ATM/HFC Network Simulator, Version 4.1, National Institute of Standards and Technology.

[11] U.T. Nguyen and I. Katzela, "A Flexible Multipoint-to-point Traffic Control Algorithm for ABR Services in ATM Networks," *Proc. of IEEE Conf. on High Performance Switching and Routing*, June 2000, pp.185-194.

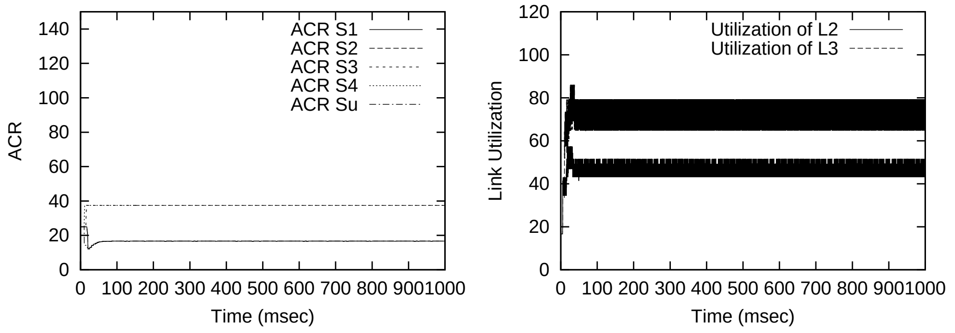
[12] D. C. W. Pao, "A Congestion Control Algorithm for Multipoint-to-Multipoint ABR Service in ATM Network," *Proc. of IEEE Conf. on High Performance Switching and Routing*, June 2000, pp.167-175.

[13] W. Ren, K.-Y. Siu, and H. Suzuki, "Multipoint-to-Point ABR Service in ATM Networks," *IEEE International Conf. on Communications*, June 1997, pp. 1185-1190.

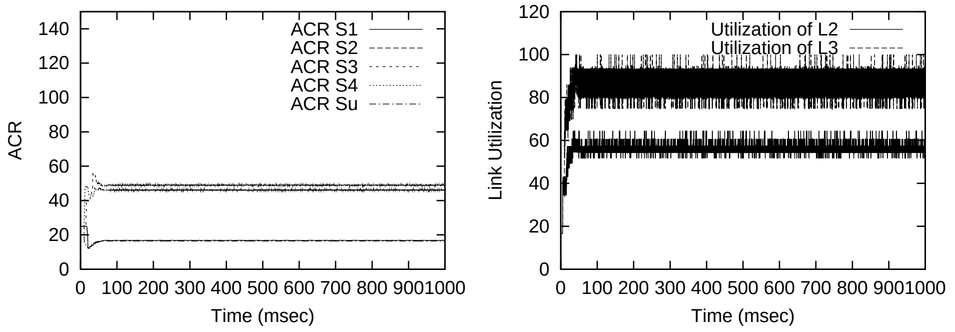
[14] W. Ren, *et al.*, "Multipoint-to-Multipoint ABR Service in ATM Networks," *Computer Networks and ISDN Systems*, vol. 30, no. 19, Oct. 1998, pp. 1793-1810.



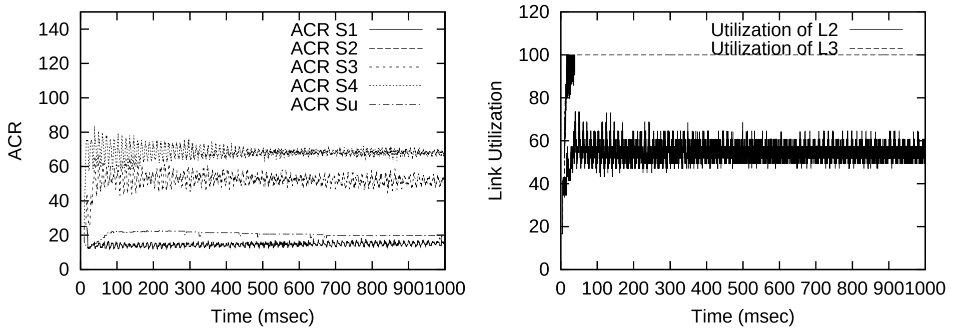
(a) Division using only branch weights: non-compliance with BAD (Configuration 2)



(b) Division using only branch weights: link under-utilization (Configuration 1)

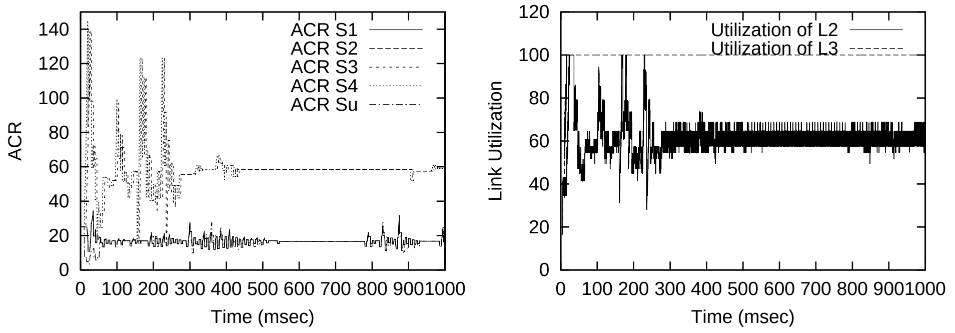


(c) Pao’s algorithm: division using both weights and aggregate CCRs of branches (Configuration 1)

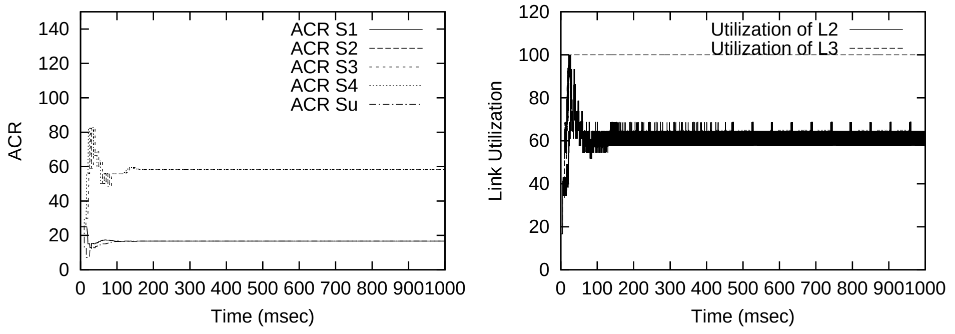


(d) Division using only aggregate CCRs of branches (Configuration 1)

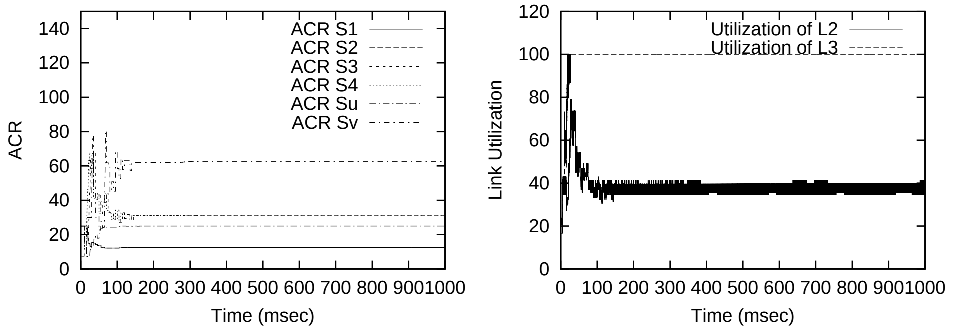
Figure 4: Simulation results for RCAs using VC weights



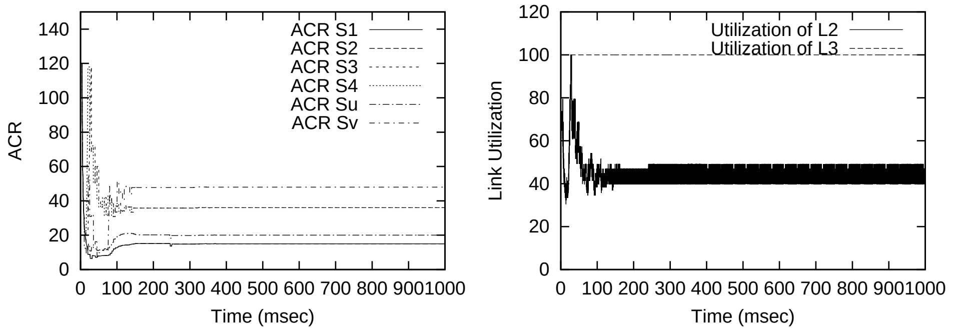
(a) FJ algorithm, Configuration 1, source-based BAD, ICR = 25 Mbps



(b) WSB algorithm, Configuration 1, source-based BAD, ICR = 25 Mbps



(c) WSB algorithm, Configuration 2, VC/source-based BAD, ICR = 25 Mbps



(d) WSB algorithm, Configuration 2, WSB BAD with mp-p source weight of 0.75, ICR = 120 Mbps

Figure 5: Simulation results for FJ and WSB algorithms