ELSEVIER

Computer Communications 24 (2001) 1673–1684

computer communications

www.elsevier.com/locate/comcom

Design and analysis of a merging algorithm for multipoint-to-point ABR service in ATM networks

Chun-Liang Lee*, Yaw-Chung Chen

Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu 30050, Taiwan, ROC

Received 8 August 2000; revised 21 December 2000; accepted 22 December 2000

Abstract

In this paper, we propose a merging algorithm, which can provide efficient support for multipoint-to-point ABR service in ATM networks. By forwarding the FRM cells belonging to the VC with the largest FRM-cell arrival rate in a merge point, the proposed algorithm can achieve better link utilization than existing merging algorithms. In addition, the proposed algorithm reduces the number of FRM cells forwarded by a merge point. As a result, it can reduce the control overhead of ABR service. Most importantly, it does not incur extra complexity in switches. We also discuss the impact of different network topologies on our algorithm. Simulation results show that the proposed algorithm is able to achieve better performance while requiring significantly fewer RM cells. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Asynchronous transfer mode; Available bit rate; Multipoint-to-point; Merging algorithm

1. Introduction

Multipoint communication is the exchange of information among multiple senders and multiple receivers. The support of multipoint communication in asynchronous transfer mode (ATM) networks is essential for applications such as audio/video conferences, distributed interactive simulations and replicated database synchronization. In recent years, issues regarding how to efficiently support multipoint-to-multipoint (mp–mp) available bit rate (ABR) service [3] have been studied extensively. A straightforward way to provide this service is combining the point-to-multipoint (p–mp) and the multipoint-to-point (mp–p) ABR services [21]. While many studies have been done on p–mp ABR service [9,12,13,15,17,19], little literature is available on mp–p ABR service.

A switch algorithm, which supports mp–p ABR service can be functionally divided into two parts: rate allocation algorithm and merging algorithm. The function of the former is to calculate a proper cell rate for a connection to utilize the network efficiently and fairly. Several existing rate allocation algorithms for mp–p ABR services [7,16,20] have been developed by modifying point-to-point (p–p) rate allocation algorithms. In mp–p ABR service, cells from different sources are merged into the same virtual connection (VC) in some merge points. Therefore, as the resource management (RM) cells belonging to different VCs leave a merge point, they are indistinguishable to the merge point when they return. A merge algorithm has the responsibility to send backward RM (BRM) cells to appropriate upstream branches when a BRM cell is received. Furthermore, it has to maintain the ratio of the number of FRM cells sent by a source to the number of received BRM cells equal to one.

In most existing merging algorithms [7,20,21], a merge point forwards all the arriving FRM cells to the outgoing link. Since these FRM cells originated from different sources, the current cell rates (CCR), which indicate the sending rates of the sources, in the FRM cells may be different. As mentioned before, these FRM cells are indistinguishable when they leave the merging point, and the downstream switches will use the CCRs to calculate the proper cell rate for this VC. If a small CCR is used, the calculated result will be a small value. This will cause unnecessary rate reductions and link under-utilization. Furthermore, it is not essential to forward all arriving FRM cells and this will be explained in Section 4.

In this paper, we propose a merging algorithm, which is able to lessen the unnecessary rate reductions. As a result, it can achieve higher link utilization than existing algorithms. The proposed algorithm also enhances the function of a merging algorithm that has not been addressed in the literature. In existing studies, the action of merging refers to the

* Corresponding author. Tel.: +886-3-5731851; fax: +886-3-5727842.
E-mail address: leecl@csie.nctu.edu.tw (C.-L. Lee).

ATM layer behavior that maps multiple VCs into a single VC. The proposed algorithm, however, really "merges" the FRM cells belonging to different VCs. In other words, fewer FRM cells are forwarded to the outgoing link. Although in our algorithm some FRM cells are merged, we still keep the basic function of a merging algorithm. That is, the ratio of the number of FRM cells sent by a source to the number of received BRM cells is equal to one. This guarantees that our algorithm does not cause the source to receive less network status carried by BRM cells. Another advantage of the proposed algorithm is that more data cells can be sent into the network because the overhead of control cells (i.e. FRM cells) is reduced.

The rest of this paper is organized as follows. In Section 2, we discuss the related issues for mp–p ABR service. Section 3 reviews two existing merging algorithms. In Section 4, we first state the motivation of this work and then describe the proposed merging algorithm. Section 5 addresses the numerical analysis of our algorithm. The analysis focuses on how much control overhead can be reduced for different network topologies. Simulation results are presented in Section 6 to evaluate the performance of the proposed algorithm. Finally, Section 7 concludes the paper and addresses the future research directions.

2. Related issues

2.1. The cell interleaving problem

In ATM networks, a switch uses the virtual path identifier (VPI) and the virtual channel identifier (VCI) carried in a cell header to forward the cell. A source sends cells with the assigned VCI and VPI, which are obtained when the connection was established, to its adjacent switch. When the switch receives a cell, it uses the VPI/VCI to lookup which output port the cell should be forwarded to, and modifies the VPI/VCI. Eventually, the cell will arrive the destination via one or more switches. The destination reassembles the arriving cells to packets based on the VPI/VCI and the end-of-packet (EOP) bit carried in each cell.

An mp–p connection can be implemented as a shared tree, as shown in Fig. 1. In such configuration, the merge point maps ATM cells from different VCs into a single VC and eventually all data streams from multiple sources are merged into a single VC. In other words, cells from different sources will arrive the destination with the same VPI and VCI.

ATM adaptation layer 5 (AAL5) is commonly used by a variety of data applications because of its simplicity and higher efficiency. However, it does not provide multiplexing identification in cells as AAL 3/4 does. If the cells belonging to different packets are interleaved, the destination cannot reassemble these packets. Therefore, it will drop these packets. This is known as the cell interleaving problem. To solve this problem, several possible solutions were proposed in [8]. In this paper, we focus on the VC merging approach since it is more scalable than the other solutions. VC merging requires merge points to guarantee the integrity of packets. When a merge point receives an incoming cell of a given packets, it needs to store the cell in a separate buffer until the last cell arrives. The last cell is identified by the end-of-packet (EOP) bit in the cell header. When the last cell is received, the merge point has to send all cells in the packet in an atomic manner. Intuitively, VC merging requires a switch to provide larger memory space for keeping the integrity of a packet. Widjaja proposed a realistic architecture for VC merging and indicated that it does not incur considerable overhead and memory requirement but can achieve higher throughput [23].

2.2. The fairness problem

For p–p connections, max–min fairness [10] is the most commonly used fairness definition. However, when mp–p connections are involved, the fairness definition becomes more complicated. Four types of fairness for mp–p ABR service have been defined: source-based fairness, VC/source-based fairness, flow-based fairness and VC/flow-based fairness. The details of these fairness definitions can be found in [8]. In this paper, we focus on the source-based fairness, which only considers the active sources. In other words, the group membership is ignored while dealing with the fairness. Therefore, an mp–p connection with n sources can be considered as n p–p connections. How to achieve a

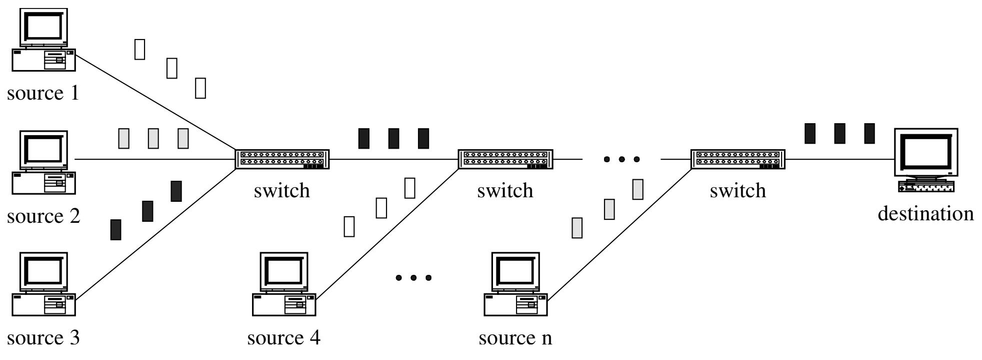

Fig. 1. A multipoint-to-point connection.

specific definition of fairness depends on the rate allocation algorithm used in switches, and this is beyond the scope of this paper.

For example, Fig. 2 shows an mp–p connection and a p–p connection. Sources A_{S1}, A_{S2} and A_{S3} are sending data to destination A_R, and source B_S is sending data to destination B_R. The capacity of each link is 150 Mbps except Link1, which has a capacity of 60 Mbps. To find the fair rate allocation for each source, we can consider the mp–p connection as three p–p connections. Therefore, the capacity of the bottleneck link, Link1, is equally distributed to A_{S1}, A_{S2} and B_S. The data rate available for A_{S3} is 110 Mbps (i.e. 150-20-20), which is the capacity of Link2 minus the data rates allocated for both A_{S1} and A_{S2}.

2.3. Rate allocation algorithm

As mentioned in Section 1, a switch algorithm which supports mp–p ABR service comprises two parts, namely rate allocation algorithm and merging algorithm. Since this paper focuses on the merging algorithm, it is necessary to choose a rate allocation algorithm to cooperate with our merging algorithm. Many point-to-point rate allocation algorithms have been proposed for ABR service [4–6,14]. However, not all of them can be used in mp–p ABR service. Based on the following reasons, we choose the explicit rate indication for congestion avoidance (ERICA) switch algorithm [14] as the rate allocation algorithm used in this paper. First, ERICA is a well-known switch algorithm which is able to provide efficient rate allocation and achieve max–min fairness. Second, ERICA has been studied and modified in many studies [1,2,22], and thus it is a more robust and well-tested algorithm. Third, it was extended in [7] to support mp–p ABR connections.

In the rest of this section, we briefly describe the ERICA algorithm for mp–p ABR service. For a complete description of this algorithm, the reader can refer to [7,11]. ERICA requires a switch to measure the load factor of each link every fixed time interval, called switch measurement interval. The load factor (z) is derived through the following equation:

$$z = \frac{\text{Input rate}}{\text{ABR capacity}}.$$

The input rate is the incoming ABR traffic measured in a measurement interval, and the capacity for ABR service is the remaining link capacity after serving those higher priority service classes, such as CBR and VBR traffic. For connection i, the switch uses a variable CCR_i to record the maximum CCR in the arriving FRM cells in the current measurement interval. Once an FRM cell of connection i is received, the switch checks the CCR field in the FRM cell and modifies CCR_i if necessary. That is,

$$CCR_i = \max(CCR_i, \text{CCR in the FRM cell}).$$

Once a BRM cell of connection i is received, the switch first checks to see if the output link is congested. If the load factor is larger than $1 + \delta$, where δ is a small fraction, the link is considered congested, and then the explicit rate (ER) field of the BRM cell will be set to CCR_i/z. This will cause the connections passing through the switch to reduce their sending rates, and keep the load factor equal to 1. To calculate the ER when the link is not congested, we have to define a variable $maxER_{pre\_intval}$, which is the maximum ER ever calculated in the previous measurement interval. The switch then chooses the larger one of CCR_i/z and $maxER_{pre\_intval}$ as the ER. This can guarantee the fairness among the connections passing through the switch. Once the ER is calculated, the switch will modify the ER field in the BRM cell if the calculated result is smaller. Otherwise, the ER field is kept the same. The following pseudo code summarizes the procedure mentioned above.

```
if ($z > 1 + \delta$) then
    $ER = CCR_i/z$;
else
    $ER = \max(CCR_i/z, maxER_{pre\_intval})$;
end if
ER in BRM cell = min($ER$, ER in BRM cell);
```

To smooth out the variations of the variable $maxER_{pre\_intval}$, the switch uses the exponential averaging to calculate

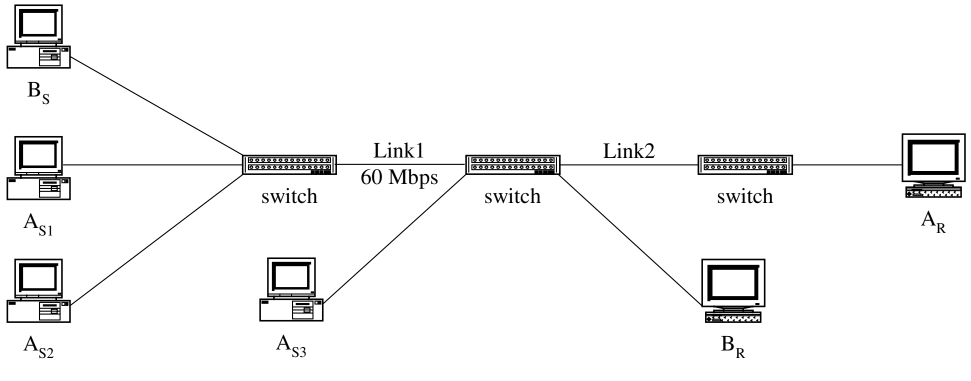

Fig. 2. An example of source-based fairness.

$maxER_{pre\_intval}$. At the end of every measurement interval, $maxER_{pre\_intval}$ is calculated using the following equation:

$$maxER_{pre\_intval} = (1 - \alpha) \times maxER_{cur\_intval} + \alpha \times maxER_{pre\_intval}.$$

The variable $maxER_{cur\_intval}$ is the maximum ER ever calculated in the current measurement interval, and the variable α is a parameter, which determines the weight of the previous result.

3. Previous works

In this section, we review two existing merging algorithms. The first algorithm was proposed by Ren et al. in Ref. [20]. In their algorithm, once a merge point receives an FRM cell from an upstream branch, it will execute the rate allocation algorithm and return a BRM cell to the upstream branch immediately. In addition, it will forward the FRM cell to the outgoing link. When the merge point receives a BRM cell, it will retrieve the required information such as ER, CI and NI, and then discard this cell. The pseudo code of the merging algorithm is as follows:

```
Upon the receipt of an FRM cell from branch $i$:
    Execute the rate allocation algorithm with this RM cell;
    Return a BRM cell to branch $i$;
    Forward this RM cell to the outgoing link;


Upon the receipt of a BRM cell:
    Execute the rate allocation algorithm with this RM cell;
    Discard this RM cell;
```

Obviously, the above algorithm guarantees that the number of FRM cells sent by a source equals to the number of BRM cells received by the source. In addition, the algorithm is simple and does not require a switch to keep any information for an mp–p connection. However, it has two major problems. First, a merge point has to generate a BRM cell once it receives an FRM cell. This will introduce extra complexity in switches. Second, a merge point that receives the network status carried by BRM cells will not send a BRM cell to the upstream unless it receives an FRM cell. Thus, the delay of the network status is sensitive to the number of merge points along the path. To overcome these problems, another algorithm was proposed by the same authors in Ref. [21], in which a merge point does not have to generate a BRM cell upon receiving an FRM cell. Instead, the merge point sends BRM cells to appropriate upstream branches once it receives a BRM cell. As a result, a *Ready* flag, which is associated with each upstream branch for an mp–p connection, is needed to control the number of BRM cells returned to the source. Whenever a merge point receives an FRM cell from a branch, it sets the corresponding *Ready* flag to 1. Once it receives a BRM cell, it sends the BRM cell to the branches whose corresponding flag equal to 1 and then resets the flag. The pseudo code of the merging algorithm is as follows:

```
Upon the receipt of an FRM cell from branch $i$:
    Execute the rate allocation algorithm with this RM cell;
    $Ready_i = 1$;
    Forward this RM cell to the outgoing link;


Upon the receipt of a BRM cell:
    Execute the rate allocation algorithm with this RM cell;
    for each upstream branch $i$ do
        if ($Ready_i = 1$) then
            Send a BRM cell to branch $i$;
            $Ready_i = 0$;
        end if
    end for
```

As compared to the first algorithm, this algorithm has the following advantages:

1. Lower complexity: as we can see in the above pseudo code, the action for sending BRM cells to the upstream links is triggered by the arrival of a BRM cell from the downstream link. Therefore, it incurs lower complexity in switches;
2. Shorter delay of network status: the network status carried in the arriving BRM cells is directly sent to the upstream branches if allowed (i.e. *Ready* flag equals to 1). As a result, this algorithm does not have the problem of network status delay as described in the first algorithm.

Due to the above advantages, this algorithm is more preferable as a merging algorithm because it is simpler and less sensitive to the number of the merge points along a path. In fact, it has also been used in other papers to study the issues regarding the mp–p ABR service [7,8]. In the rest of the paper, we only consider this algorithm and denote it as Ren's algorithm.

4. Proposed merging algorithm

4.1. Motivation

Although Ren's algorithm is better than the first algorithm we mentioned, it has the following two drawbacks. First, it forwards all the arriving FRM cells to the downstream network. These FRM cells from different VCs are indistinguishable to the downstream switches when they leave the merge point. Most rate allocation algorithms, including ERICA, use the CCR field in FRM cells to calculate the allowed cell rate for the VC. Since the CCRs of different VCs are different, if a small CCR is used to calculate the allowed cell rate by the downstream switches, this may cause unnecessary rate reductions.

Second, since the FRM cells are indistinguishable when they leave the merge point, it is unnecessary for a merge point to forward all the arriving FRM cells. In ATM networks, RM cells are used to convey the network status to sources for the closed-loop flow control. This means RM cells do not carry users data. In other words, they are control overhead, which should be kept as few as possible. However, reducing the amount of RM cells does not always benefit the throughput since it also reduces the frequency of network status received by a source. As a result, the source may response to the change of network status slowly, which leads to worse throughput. In the rest of this section, we will show that it is possible to forward less FRM cells but can still generate enough BRM cells sent to the source. More importantly, it can achieve better efficiency.

Consider the scenario shown in Fig. 3. Two VCs are merged into one VC in a merge point. The variables r_1 and r_3 represent the FRM cell rates of these two VCs while variable r_2 and r_4 represent the BRM cell rates generated by the merge point for these two VCs. The variable r_6 denotes the rate of the BRM cells from the downstream switch.

As in Ren's algorithm there is only one bit used for each branch to record the status of the arriving FRM cells, we have the following equations:

$$r_2 = \min(r_1, r_6) \tag{1}$$

$$r_4 = \min(r_3, r_6) \tag{2}$$

Recall that in Ren's algorithm, a merge point forwards every FRM cell to the outgoing link. Hence, we have $r_5 = r_1 + r_3$. Suppose that the transmission rates of sources do not vary dramatically. The values of r_5 and r_6 will be very close (i.e. $r_5 \cong r_6$). Therefore, Eqs. (1) and (2) can be rewritten as

$$r_2 = \min(r_1, r_1 + r_3) = r_1 \tag{3}$$

$$r_4 = \min(r_3, r_1 + r_3) = r_3 \tag{4}$$

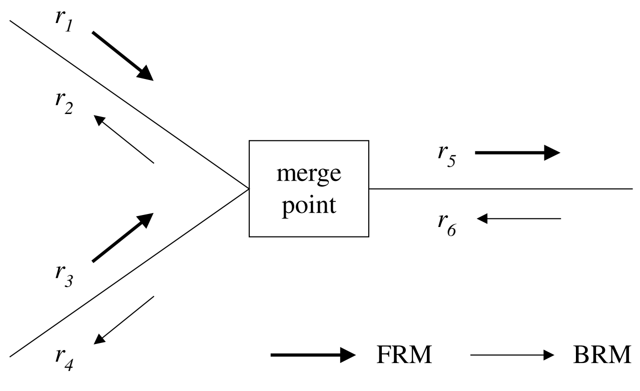

Fig. 3. A simple merge point scenario.

As shown in Eqs. (3) and (4), for each branch, the merge point keeps the ratio of the number of FRM cells to that of the BRM cells close to 1, and this is one of the required functions of a merge point.

Our initial idea is that if the merge point forwards the FRM cells of the VC with the rate of max(r_1, r_3), we have

$$r_6 \cong r_5 = \max(r_1, r_3) \tag{5}$$

Substituting Eq. (5) into Eqs. (1) and (2), we have the following equations:

$$r_2 = \min(r_1, \max(r_1, r_3)) = r_1 \tag{6}$$

$$r_4 = \min(r_3, \max(r_1, r_3)) = r_3 \tag{7}$$

Therefore, the merge point can forward fewer FRM cells but can still generate enough BRM cells for all branches. For a switch, which merges more than two VCs, we can forward the FRM cells of the VC with the largest cell rate to achieve the same purpose.

4.2. Proposed algorithm

Since it is complex and hard to determine the cell rate of each VC, we try to replace the flag with a counter, denoted as *cntr*, for each branch to achieve the same purpose. In the following, we present our proposed algorithm. When a merge point receives an FRM cell from an upstream branch, it increases the corresponding counter by 1. If the counter value is the largest among all counters after the increment, the merge point will send the FRM cell to the outgoing link. Otherwise, it will discard the FRM cell. When the merge point receives a BRM cell, it sends the BRM cell to the upstream braches whose corresponding counters are larger than 0. Before sending a BRM cell, the merge point will execute the rate allocation algorithm to calculate the explicit rate for the VC and put the rate in the BRM cell. The pseudo code of the proposed algorithm is as follows:

```
Upon the receipt of an FRM cell from branch $i$:
  Execute the rate allocation algorithm with this RM cell;
  $cntr_i$++:
  if ($cntr_i$ is the largest among all branches) then
    Forward this RM cell to the outgoing link;
  else
    Discard this RM cells;
  end if


Upon the receipt of a BRM cell:
  for each upstream branch $i$ do
    if ($cntr_i > 0$) then
      Execute the rate allocation algorithm;
      Send a BRM cell to branch $i$;
      $cntr_i$− − ;
    end if
  end for
```

As compared with the existing algorithms, our algorithm incurs subtle extra complexity both in space and time. First, the memory size required by each branch is the size of a counter, which is determined by the maximum distance from a merge point to the destination and the maximum cell rate of a VC. Practically, one or two bytes should be enough. With the price of memory getting cheaper and cheaper, the cost of counters can be ignored. If the counter size is too small but the transmission path is too long, the counter may overflow and cause our algorithm to misbehave. The problem, however, can be overcome easily by implementing a non-overflow addition operation.

Second, in our algorithm, a merge point has to determine whether the value of a counter is the largest among all the branches. The time complexity of finding the largest one in a set of values is O(n), where n denotes the number of branches. Obviously, n is no larger than the number of ports in a switch, and it is usually a small value. Therefore, our algorithm does not incur considerable complexity.

4.3. Complexity reduction

As discussed in Section 4.2, the proposed algorithm is simple and cost effective. However, in order to make the proposed algorithm more practical, we present an improved approach whose time complexity is only O(1). We introduce a variable *MAX* for each mp–p connection. The variable *MAX* is used to keep the largest counter value and it is initially set to 0. Once a merge point receives an FRM cell from a branch, it will check whether the value of the corresponding counter is equal to *MAX* before increasing the counter. If these two values are identical, that means the counter value will become the largest one after the increment. Therefore, this FRM cell will be forwarded to the outgoing link and both the counter and *MAX* are increased by 1. When the merge point receives a BRM cell, it will perform the same operation as our first algorithm. In addition, *MAX* is decreased by 1. The improved algorithm is shown as follows:

```
Upon the receipt of an FRM cell from branch $i$:
  Execute the rate allocation algorithm with this RM cells;
  if ($cntr_i = MAX$) then
    $cntr_i{+}{+}$;
    $MAX{+}{+}$;
    Forward this RM cell to the outgoing link;
  else
    Discard this RM cell;
  end if

Upon the receipt of a BRM cell:
  MAX$-\,-$;
  for each upstream branch $i$ do
    if ($cntr_i > 0$)
      Execute the rate allocation algorithm;
      Send a BRM cell to branch $i$;
      $cntr_i -\,-$;
    end if
  end for
```

5. Numerical analysis

In this section, we analyze the quantity of RM cells that can be reduced by our proposed algorithm. First, the analysis is based on a single merge point. Then we can use the analysis result to calculate the total number of reduced RM cells for an mp–p connection. We also discuss the performance of our algorithm under different network topologies. Especially, we are interested in the network topologies in which our algorithm will have either the best or the worst performance. Since it is hard to analyze the proposed algorithm during the bandwidth contention period, in the following discussion we assume the system is in steady state.

Given an mp–p connection with n sources, as stated in Section 1, the data streams from these n sources will be merged in some merge points in the network. Let R_i denote the percentage of FRM cells reduced by our algorithm in a merge point, which has i upstream sources. Assume that these sources are src_1, src_2,…, src_i. Let r_i be the rate of FRM cells sent from src_i. Recall that Ren's algorithm forwards all the received FRM cells to the downstream network. On the contrary, our proposed algorithm forwards the FRM cells belonging to the VC with the largest FRM-cell arrival rate. Therefore, we have the following equation:

$$R_i = \frac{\sum_{j=1}^{i} r_j - \max(r_1, r_2, \ldots, r_i)}{\sum_{j=1}^{i} r_j} = 1 - \frac{\max(r_1, r_2, \ldots, r_i)}{\sum_{j=1}^{i} r_j}. \tag{8}$$

If the merge point is the bottleneck and there is no constrained source, we have

$$r_1 = r_2 = \cdots = r_i. \tag{9}$$

Applying Eq. (9) to Eq. (8), we have the following equation, which is the maximum percentage of FRM cells that can be reduced by our algorithm.

$$R_i = 1 - \frac{r_1}{\sum_{j=1}^{i} r_1} = 1 - \frac{1}{i}. \tag{10}$$

After deriving the percentage of FRM cells reduced, we can derive the amount of saved bandwidth of a link as follows. Let BW_{RM} and BW_{ABR} denote the link capacity used by RM cells and ABR traffic, respectively. Then the amount of saved bandwidth (BW_{saved}) can be derived as

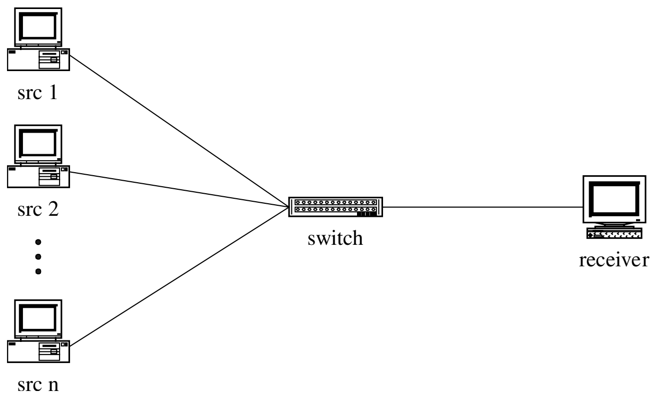

Fig. 4. Configuration of the best case.

follows:

$$BW_{saved} = BW_{RM} R_i. \tag{11}$$

In ABR service, the RM cells are sent by source after *Nrm*-1 data cells, where *Nrm* is a parameter with a default value of 32. In other words, the ratio of the number of RM cells to the number of data cells is 1/(*Nrm*-1). Therefore, we have the following equation:

$$BW_{RM} = \frac{BW_{ABR}}{Nrm}. \tag{12}$$

Substituting Eq. (12) into Eq. (11), we have

$$BW_{saved} = \frac{BW_{ABR}}{Nrm} R_i. \tag{13}$$

So far we have analyzed the amount of saved bandwidth in a single merge point. The total amount of saved bandwidth in the network can be calculated by considering all the merge points. In the rest of this section, we discuss both the best case and the worst case of our proposed algorithm from the aspect of network topology. First of all, we consider the network topology in which our algorithm will have the best performance. Same as that specified in the previous section, we assume that there is a mp–p connection with n sources. Since our algorithm merges VCs in a merge point, the best case would be that all VCs are merged in only one merge point. Therefore, the least amount of FRM cells will flow in the network. Fig. 4 shows the network configuration, in which the FRM cells in $n-1$ out of n VCs would be merged. Because there is only one merge point, the percentage of FRM cells reduced can be derived through Eq. (8) where $i = n$. The largest value of R_i will appear in the case that all sources have the same transmission rate, and it can be derived through Eq. (10).

Contrary to the best case, the network topology of the worst case would be that n VCs are merged with the largest number of merge points, that is, $n-1$. Fig. 5 shows the network configuration of the worst case. When r_1 or r_2 is the maximum rate among all source transmission rates, our algorithm reduces least amount of FRM cells. In other words, one of the following two equations holds in the worst case

$$r_1 = \max(r_i) \;\; 1 \le i \le n \tag{14}$$

$$r_2 = \max(r_i) \;\; 1 \le i \le n. \tag{15}$$

Without loss of generality, we assume that Eq. (14) holds in the worst case. The percentage of FRM cells ($R_{network}$) totally reduced can be derived as follows:

$$R_{network} = \frac{\sum_{i=1}^{n}(n-i+1)r_i - r_1 - (n-1)r_1}{\sum_{i=1}^{n}(n-i+1)r_i - r_1} \quad \forall i \ge 2. \tag{16}$$

In the worst-case configuration, the upper bound of $R_{network}$ can be obtained when Eq. (9) holds. This means that the outgoing link of the last merge point is the bottleneck link, and it is fairly shared by n sources. If the fair rate share of each source is M, we can derive the upper bound of $R_{network}$

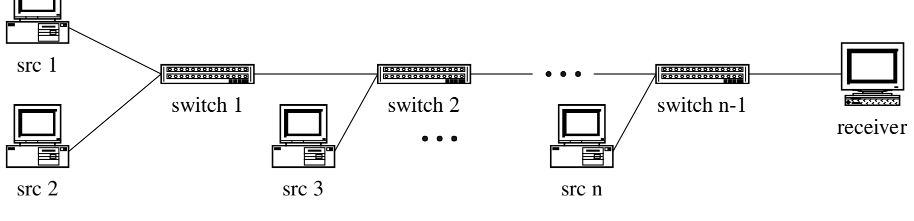

Fig. 5. Configuration of the worst case.

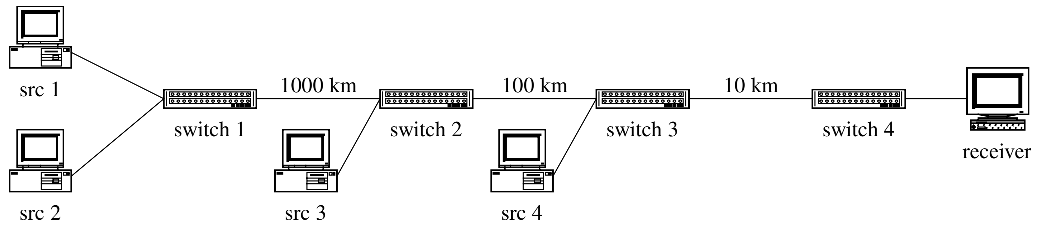

Fig. 6. Simulation configuration.

through the following equation:

$$R_{network} = \frac{\sum_{i=1}^{n}(n-i+1)M - M - (n-1)M}{\sum_{i=1}^{n}(n-i+1)M - M} = \frac{n}{n+2} \quad \forall i \geq 2. \tag{17}$$

6. Simulation results

To evaluate the performance of the proposed algorithm, we present the simulation results in this section. The simulation configuration used is shown in Fig. 6, in which each link has a capacity of 150 Mbps. The length of each link connecting adjacent switches is shown in the figure. The other access links are assumed to be 100 m long. There is a mp–p connection, which has four sources, *src*1, *src*2, *src*3 and *src*4, sending data to the same destination. In our simulation, we assume that the sources always have data to transmit, and each cell stream is generated according to the allowed cell rate of the corresponding source. The simulation time is 300 ms, which is long enough for the simulation process to reach the steady state. The parameters of ABR source and ERICA are shown in Tables 1 and 2, respectively. The values of ERICA parameters are set according to the guidelines stated in Ref. [11].

In the rest of this section, we compare the performance of our proposed algorithm to Ren's algorithm in four aspects: the allowed cell rate (ACR) of each source, the queue occupancy of each switch, the number of FRM cells received by the destination, and the utilization of the bottleneck link.

Figs. 7 and 8 show the ACR of each source in Ren's and our algorithms, respectively. Since the ICR of each source is set to 150 Mbps (i.e. PCR), all switches will experience congestion in the beginning of the simulation. After the sources reduce their rates, they will start to contend for the bandwidth of the bottleneck link. Therefore, Figs. 7 and 8 can demonstrate the basic behavior of Ren's and our algorithms. As shown in the figures, the time required for sources to reach the stable state is approximately 120 ms in Ren's algorithm, while it is slightly longer in our algorithm, which is about 130 ms. Although a little more time is required for our algorithm to reach the steady state, we can see that the ACR of each source in both algorithms are very close to the fair rate after time $T = 100$ ms. This shows that our algorithm does not degrade the efficiency as contrast to Ren's algorithm, while using much less RM cells. In addition, in Ren's algorithm, the ACRs of *src*1 and *src*2 decrease to very low values, say less than 5 Mbps, during several time periods from $T = 20$ to 50 ms, while in our algorithm they are kept larger than 20 Mbps. This is because Ren's algorithm forwards all received FRM cells belonging to different sources to the downstream node. As we can see in Fig. 7, during the time period from $T = 20$ to 50 ms, the ACRs of *src*3 and *src*4 are very small, and the information is indicated in the CCR fields of the FRM cells sent from *src*3 and *src*4. Once *swtich*3 uses the information to calculate the new explicit rate of the VC, it will get a wrong result, which causes *src*1 and *src*2 to reduce their sending rates unnecessarily. On the contrary, the proposed algorithm does not have this problem. In our algorithm, a merge point forwards the FRM cells, which belong to the VC with largest FRM-cell arriving rate. In other words, only the FRM cells from the source with largest ACR will be forwarded by the merge point. Therefore, our algorithm can avoid the unnecessary rate reductions.

Figs. 9 and 10 show the queue length in each switch under different algorithms. As seen in the figures, *switch*1 has the largest queue length since *src*1 and *src*2 have the largest round-trip times. However, *switch*3 is the slowest one whose queue length is reduced to zero because it is the bottleneck node. Comparing these two figures, the time required to reduce the queue length to zero is slightly longer

Table 1
Source parameters

Parameter	Value	Meaning
MCR	1 Mbps	Minimum cell rate
PCR	150 Mbps	Peak cell rate
ICR	150 Mbps	Initial cell rate
RIF	0.0625	Rate increase factor

Table 2
Switch parameters

Parameter	Value
α	0.1
δ	0.1
Target utilization	0.9
Switch measurement interval	1 ms

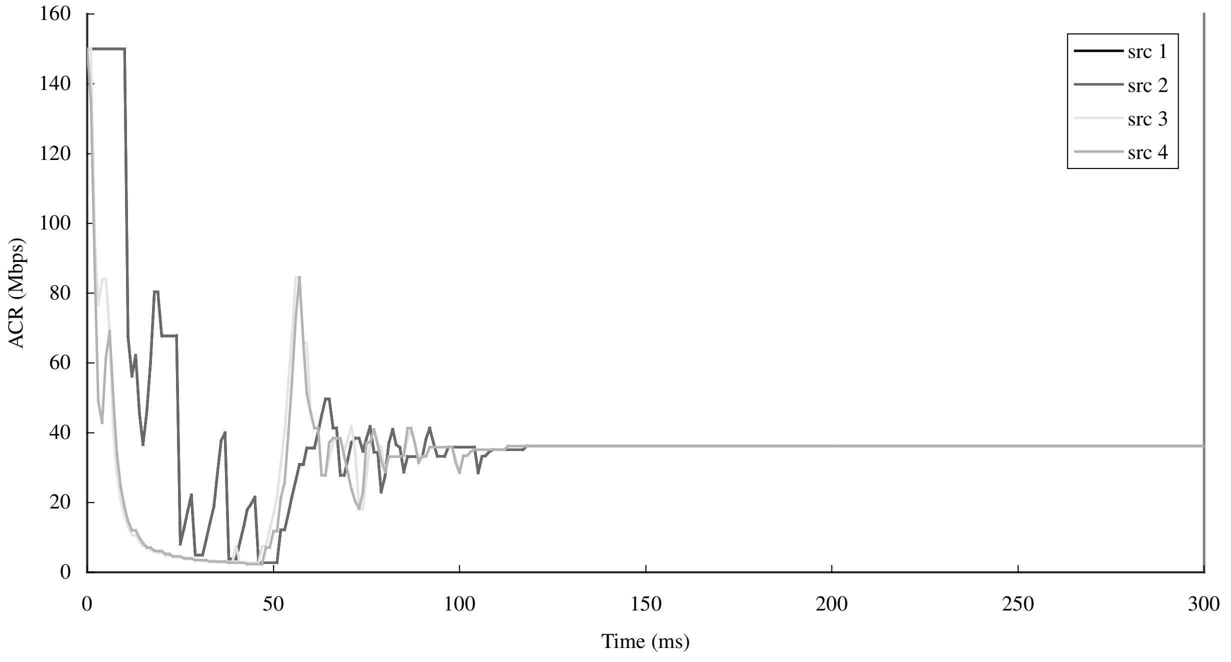

Fig. 7. ACRs in Ren's algorithm.

for our algorithm than that for Ren's algorithm. The main reason is that Ren's algorithm causes unnecessary rate reductions; therefore, more bandwidth is used to drain the queue. This does not mean that our algorithm fails to control the queue length efficiently. In fact, our algorithm provides higher link utilization than Ren's algorithm. This will be discussed later.

Fig. 11 shows the number of received FRM cells in the destination for Ren's and our algorithms. The results are expressed as a percentage of the number of received FRM cells in Ren's algorithm so we can see that the results for Ren's algorithms are all 100%. As stated in our simulation configuration, the link between *switch*3 and *swtich*4 is the bottleneck link, where all data streams pass through. Therefore, the link capacity is fairly allocated to these four sources. By Eq. (10) where i equals to 4, the percentage of FRM cells reduced by our algorithm is 75% as compared to Ren's algorithm. As seen in Fig. 9, when the bandwidth contention period (i.e. from time $T = 0$ to 60 ms) is over, the number of received FRM cells in our algorithm is kept at

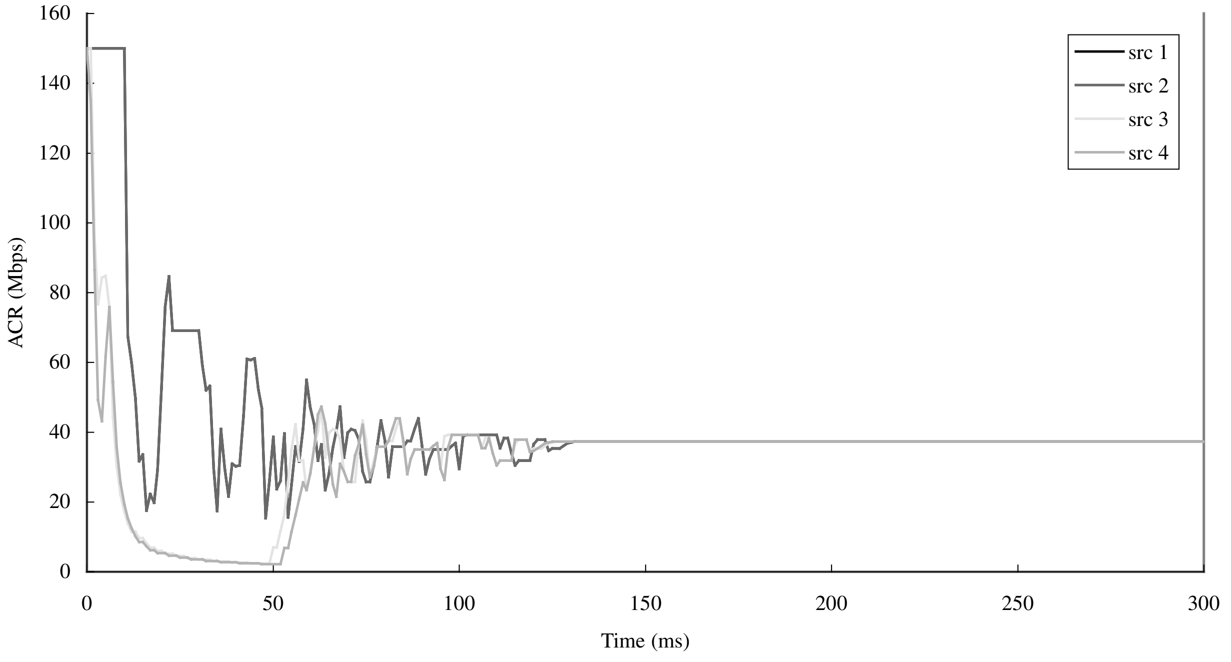

Fig. 8. ACRs in our algorithm.

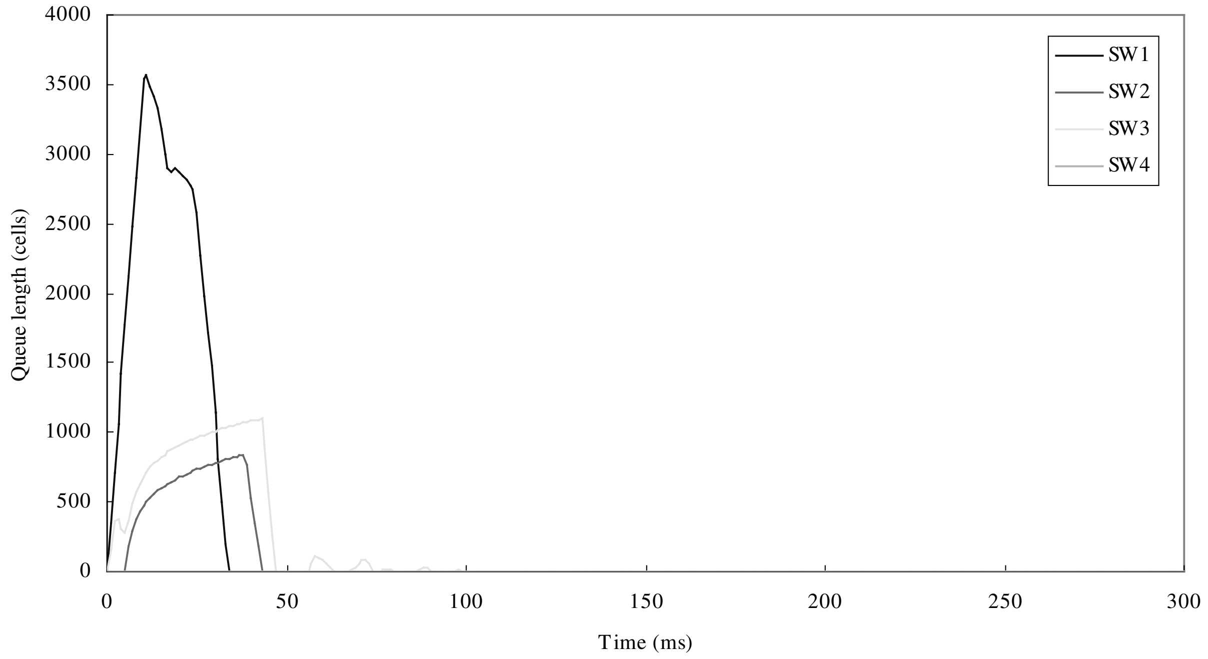

Fig. 9. Queue lengths in Ren's algorithm.

25% of that in Ren's algorithm, which is consistent with our analysis. Even in bandwidth contention period, our algorithm still reduces considerable amount of FRM cells except at time $T = 60$ ms. This is because Ren's algorithm reduces the ACRs of *src*1 and *src*2 unnecessarily as discussed earlier, and the queue of *swith*3 is drained off before the ACRs increase. Therefore, during the time period from $T = 50$ to 60 ms the number of cells passing through *swith*3 in Ren's algorithm is much less than that in our algorithm. In this case, the destination receives more FRM cells in our algorithm than it does in Ren's algorithm. For further verification, Fig. 12 illustrates the utilization of the bottleneck link. As we expected, the link utilization in Ren's algorithm degrades to approximately 20% around time $T = 50$ ms, while it keeps at a higher value in our algorithm.

7. Conclusions and future work

In this paper, we have proposed a merging algorithm,

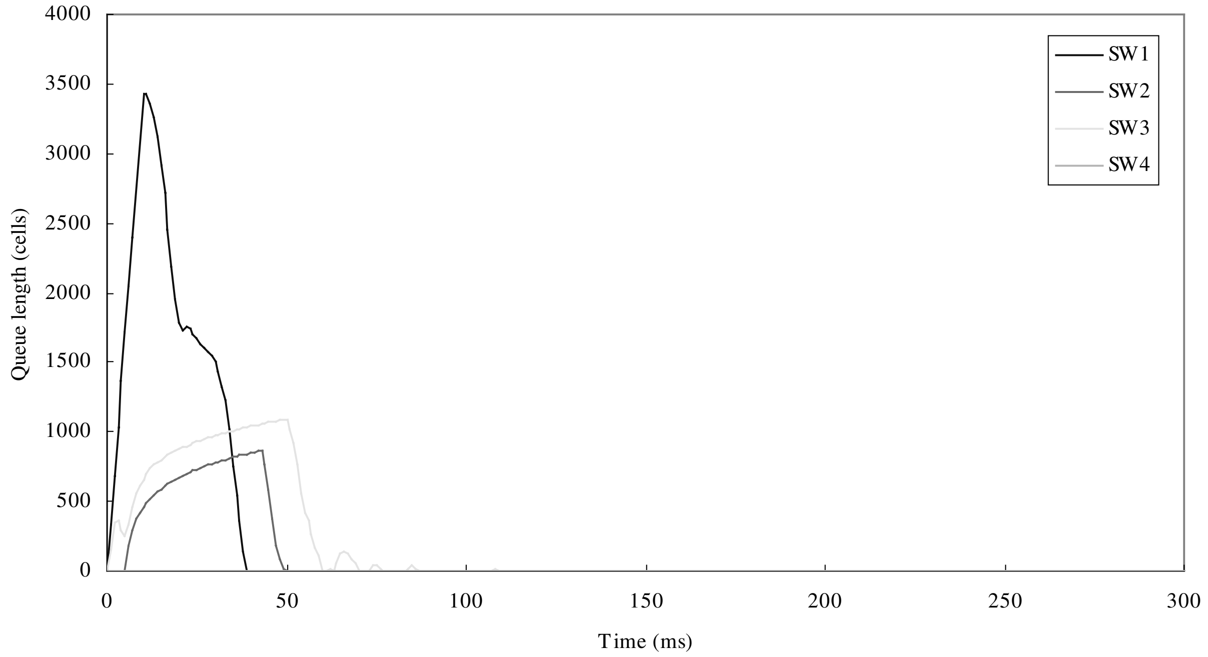

Fig. 10. Queue lengths in our algorithm.

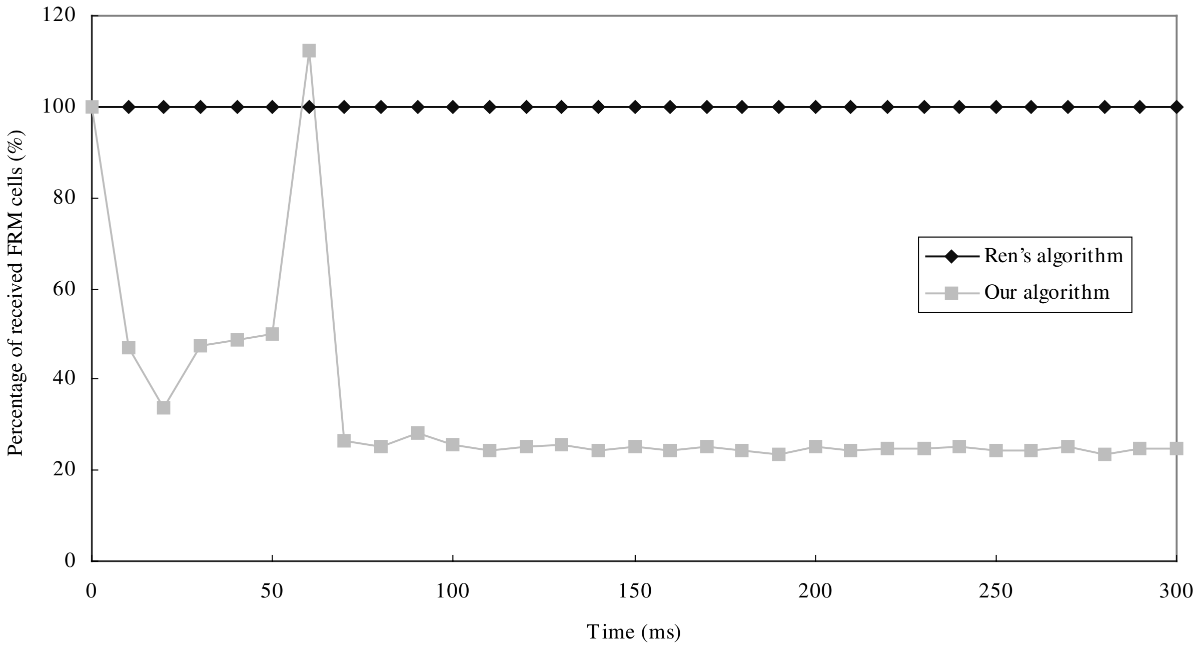

Fig. 11. Number of received FRM cells.

which has rarely been addressed in the literature. By forwarding the FRM cells belonging to the VC with the largest FRM-cell arrival rate, the proposed algorithms are able to decrease the unnecessary rate reductions. As a result, link utilization can be improved as compared to existing algorithms. To deal with the arrival of an FRM cell or a BRM cell, the running time of the proposed algorithm is O(1), which means that our algorithm does not incur extra complexity in the switch. In addition, the proposed algorithm can reduce the quantity of FRM cells sent to the downstream network. We have also analyzed how much control overhead can be reduced under different network topologies.

As mentioned in Section 2.2, this paper focuses on the source-based fairness. Issues regarding other fairness definitions still need further study. As new fairness definitions have been proposed [18], how to design a good rate allocation algorithm and merging algorithm for different fairness definitions, even for all of them, is a challenging work. With the lessons learnt from this work, we are

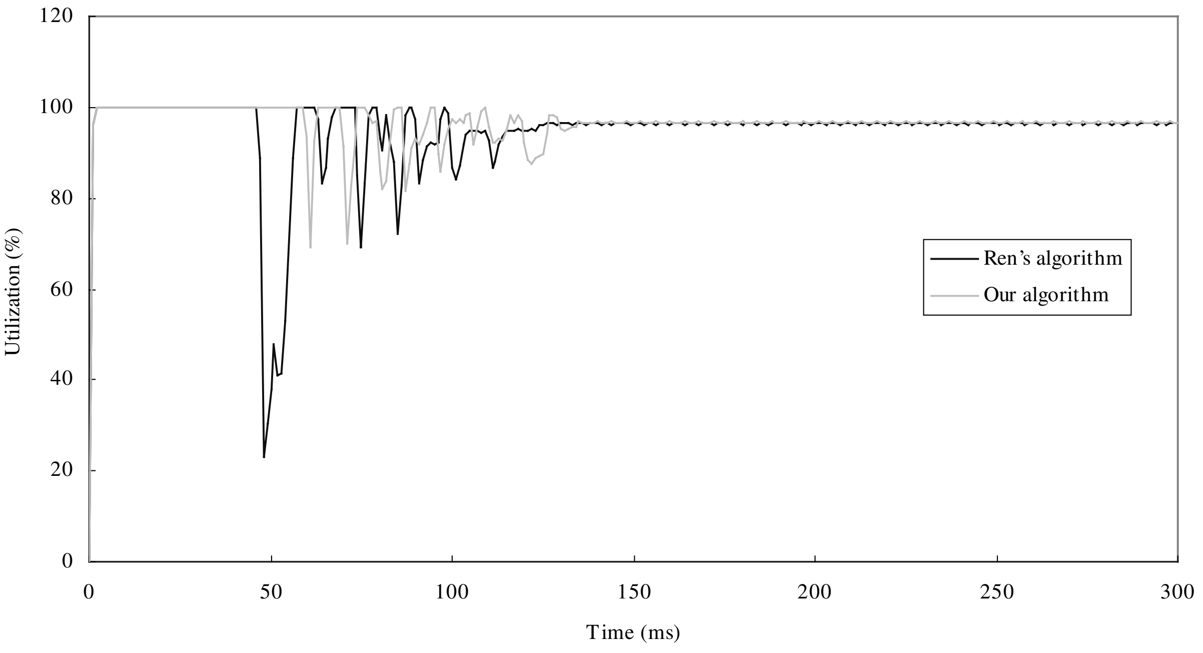

Fig. 12. Utilization of the bottleneck link.

currently working on designing a rate allocation algorithm, which cooperates with the proposed merging algorithm to support more than one types of fairness.

Acknowledgements

The authors would like to thank the anonymous reviewers who provided helpful feedback on the manuscript.

References

[1] Y. Afek, Y. Mansour, Z. Ostfeld, Phantom: a simple and effective flow control scheme, Proc. ACM SIGCOMM (1996) 169–182.

[2] A. Arulambalam, X. Chen, N. Ansari, Allocating fair rates for available bit rate service in ATM networks, IEEE Commun. Mag. 34 (11) (1996) 92–100.

[3] The ATM Forum Technical Committee. ATM Forum traffic management specification version 4.1, ATM Forum AF-TM-0121.000, March 1999.

[4] A. Charny, D.D. Clark, R. Jain, Congestion control with explicit rate indication, Proc. Int. Conf. Commun. 3 (6) (1995) 1954–1963.

[5] Y.C. Chen, C.T. Chan, S.C. Hu, On the effective traffic control of ABR services in ATM networks, IEICE Trans. Commun. 2 (2) (1998) 417–430.

[6] F.M. Chiussi, A. Arulambalam, Y. Xia, X. Chen, Explicit rate ABR scheme using traffic load as congestion indicator, Proc. 6th Int. Conf. Comp. Commun. Networks 9 (1997) 76–84.

[7] S. Fahmy, R. Jain, R. Goyal, B. Vandalore, A switch algorithm for ABR multipoint-to-point connections, ATM Forum Contribution 97-1085R1, December 1997.

[8] S. Fahmy, R. Jain, R. Goyal, B. Vandalore, Fairness for ABR multipoint-to-point connections, Proc. SPIE Symp. Voice Video Data Commun., vol. 3530, Conference on Performance and Control of Network Systems II, (11) (1998) 131–142.

[9] S. Fahmy, R. Jain, R. Goyal, B. Vandalore, S. Kalyanaraman, S. Kota, P. Samudra, Feedback consolidation algorithms for ABR point-to-multipoint connections in ATM networks, Proc. IEEE INFOCOM 3 (1998) 1004–1013.

[10] J.M. Jaffe, Bottleneck flow control, IEEE Trans. Commun. 29 (7) (1980) 954–962.

[11] R. Jain, S. Fahmy, S. Kalyanaraman, R. Goyal, ERICA switch algorithm: a complete description, ATM Forum 96-1172, August 1996.

[12] T. Jiang, M. Ammar, E.W. Zegura, Inter-receiver fairness: a novel performance measure for multicast ABR sessions, Proc. ACM SIGMETRICS 6 (1998) 202–211.

[13] T. Jiang, E.W. Zegura, M. Ammar, Improved consolidation algorithms for point-to-multipoint ABR service, Proc. IEEE ATM Workshop (1998) 195–201.

[14] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, B. Vandalore, The ERICA switch algorithm for ABR traffic management in ATM networks, IEEE/ACM Trans. Networking 8 (1/2) (2000) 87–98.

[15] D.H. Kim, Y.B. Park, J.K. Kim, Point to multipoint ABR flow control in ATM networks, Proc. IEEE Conf. High Perform. Switching and Routing 6 (2000) 177–184.

[16] W.M. Moh, Y. Chen, Design and evaluation of multipoint-to-point multicast flow control, Proc. SPIE Conf. Perform. Control Network Systems II 11 (1998) 143–154.

[17] W.M. Moh, Y. Chen, B. Niizawa, Branch-point algorithms for multicasting ATM ABR protocols, Proc. IEEE Workshop ATM 5 (1998) 202–211.

[18] U.T. Nguyen, I. Katzela, A flexible multipoint-to-point traffic control algorithm for ABR services in ATM networks, Proc. IEEE Conf. High Perform. Switching Routing (2000) 185–194.

[19] W. Ren, K.Y. Siu, H. Suzuki, On the performance of congestion control algorithms for multicast ABR service in ATM, Proc. IEEE ATM Workshop (1996).

[20] W. Ren, K.Y. Siu, H. Suzuki, Multipoint-to-point ABR service in ATM networks, Proc. Int. Conf. Commun. 3 (6) (1997) 1185–1190.

[21] W. Ren, K.Y. Siu, H. Suzuki, M. Shinohara, Multipoint-to-multipoint ABR service in ATM, Computer Networks ISDN Systems 30 (10) (1998) 1793–1810.

[22] D. Tsang, W. Wong, A new rate-based switch algorithm for ABR traffic to achieve max–min fairness with analytical approximation and delay adjustment, Proc. IEEE INFOCOM 3 (1996) 1174–1181.

[23] I. Widjaja, A.I. Elwalid, Performance issues in VC-merge capable switches for multiprotocol label switching, IEEE JSAC 17 (6) (1999) 1178–1189.