A Simple Timeout Algorithm for Point-to-Multipoint ABR Service

Wei Kuang Lai, Chien Ting Chen, and Chilin Li

Abstract: **The ABR point-to-multipoint connection is now playing a more important role than before. Many consolidation algorithms have been proposed to solve the consolidation noise problem and the slow transient response problem. But few timeout algorithms are proposed to handle the non-responsive branches for the multicast connections. Chen's algorithm needs exchanging control messages between switches [9]. Besides, it may mistake a responsive branch as a non-responsive branch because of fast changes in source rates, which causes wrong information in BRM cells and may lead to network congestion and data losses in the responsive branch. We propose a simple timeout algorithm which can handle the non-responsive branches without exchanging message between switches. The timeout value for each switch is computed locally. Simulation results show that the proposed timeout algorithm can efficiently handle the non-responsive branches and utilize the available bandwidth within a small period of time. In addition, our algorithm could handle the situation when the source rates change quickly.**

Index Terms: **Multicast, ATM, branch.**

I. INTRODUCTION

The point-to-multipoint, or multicast, ABR service is important for many emerging data applications. These applications include audio and video conferences, video on demand, distance learning, tele-metering, distributed games, server and replicated database synchronization, advertising, searching, and data distributions.

In ABR point-to-point connections, the source sends cells at the minimum speed that can be supported by all the switches on the path from the source to the destinations. It uses a close-loop flow control to adjust the source sending rate according to network conditions.

In ABR point-to-multipoint flow control, we assume that a multicast tree has been created from the source to destinations in this paper. For better understanding and reference, we list related terms of this paper and their meanings in Table 1. Since the destinations send BRM (Backward Resource Management) cells back to the source, the branch points have to consolidate the information such as ER (Explicit Rate), CI (Congestion Indication), and NI (No Increase) from the downstream nodes. There are several problems. First, there may have been more than one feedback. Among them, we have to decide which one is up-to-date or important, the consolidation noise problem. We also have to avoid slow response because of many branches or long delays from some branches, the slow transient response problem. In addition, there may have some non-responsive branches which increase the response time and form an interesting problem. This kind of problem is named the non-responsive branch problem in this paper. Many consolidation algorithms (branch algorithms) have been proposed to solve the consolidation noise problem and the slow response transient problem [3]–[8].

Roberts [3] proposed that a branch point, usually a switch, should return a BRM cell to its upstream node whenever it receives an FRM (Forward Resource Management) cell. The ER in the BRM cell is the minimum of the ER in the last FRM cell and all ERs reported by downstream branches since the last BRM cell was generated. Tzeng and Siu [4] proposed a slightly more conservative algorithm than Roberts' algorithm. The branch point can only send a BRM cell to its upstream node when it receives a FRM cell and it has received at least one BRM cell since the last BRM cell is sent to its upstream node. Ren, Siu, and Suzuki [5] argued that the branch point should avoid the heavy processing overheads of RM cells. They proposed that the branch point does not need to initiate the generation of BRM cells, but simply forwards selected BRM cells received from downstream nodes, after modifying the contents. The "wait-for-all" algorithm is the one that can totally reduce the consolidation noise [5]. It allows the branch point to send a BRM cell to its upstream node only when it received all the feedback from downstream branches. The "wait-for-all" algorithm has a possible problem. If there is no timeout mechanism for each branch, the "wait-for-all" algorithm will not be able to respond to the changes of the network conditions whenever any branch becomes non-responsive.

However, few algorithms are proposed to solve the non-responsive branch problem. Chen [9] proposed a dynamic timeout algorithm to handle non-responsive branches for above algorithms such as the "wait-for-all" algorithm. But the complexity of implementation is too high. Besides, the algorithm will mistake a responsive branch as a non-responsive branch when the source rates change quickly. When a branch is responsive but the algorithm mistakes it as non-responsive, the information carried in the BRM cell may be incorrect. Moreover, the source may then send data at a rate which exceeds the transmission rate of the slow branch. It can lead to network congestion in the branch and data transmission losses for upper layer applications. When a branch is non-responsive but the algorithm mistakes it as responsive, the responsive time will be prolonged. Since the former mistake is less desirable than the later, we should try to avoid the former kind of mistake. We propose a new timeout algorithm for the ABR multicast connection with lower cost of implementation. The new timeout algorithm would not mistake a responsive branch as a non-responsive branch when source rates

Manuscript received October 3, 2001; approved for publication by Tony Lee, Division III Editor, November 9, 2002.

The authors are all with department of computer science and engineering, National Sun Yat-Sen Universality, Taiwan 804, email: wklai@cse.nsysu.edu.tw.

Table 1. The terms used in this paper and their meanings.

BRM (Backward Resource Management) cell	A resource management cell sent from the switch or destination to the source.
FRM (Forward Resource Management) cell	A resource management cell sent from the source to the destination.
STV (Switch Time Out Value)	Each switch maintains a STV which is the largest timeout among the timeouts of branches. The STV is used to update the timeout value of the switch's upstream switches.
FRTT (Fixed Round Trip Time)	The sum of the fixed and propagation delays from the switch to the leaf and back, by using the ABR signaling messages.
Trm	The default value of Trm is 100 milliseconds and it provides an upper bound between two FRM cells for an active source.
Nrm	The default value is 32 cells which is the maximum number of cells a source may send for each forward RM cell.
NFRM	A counter maintained by a switch to store the number of FRM cells received between two unmarked BRM cells.
ICR (Initial Cell Rate)	The rate at which a source should send initially and after idle period.
MCR (Minimum Cell Rate)	The rate at which the source is always allowed to send.
PCR (Peak Cell Rate)	The rate at which the source may never exceed.
CCR (Current Cell Rate)	The rate at which the source sends currently.
ER (Explicit Rate)	The rate which the source is allowed to send in the network.
CI (Congestion Indication)	Used to notice the source that there is congestion in the network.
NI (No Increase)	Used to notice the source not to increase its rate.

change often. The rest of the paper is organized as follows. Section II is an introduction of Chen's algorithm. We then propose a simple timeout algorithm for solving the non-responsive branch problem in Section III. We also list the differences between our algorithm and Chen's algorithm in Section III. Simulation results and comparison of our algorithm and Chen's algorithm for different background CBR traffic are shown in Section IV. Conclusions are in Section V.

II. CHEN'S ALGORITHM

In Chen's algorithm, each switch will maintain a timeout value of each branch. A timeout value of a branch is the longest waiting time for all its leaves to return a BRM cell. Each switch will also maintain a Switch Timeout Value (STV) which is the largest timeout among the timeouts of branches. A leaf is called an rm-inactive leaf if the leaf has not received an FRM cell. An rm-inactive branch is a branch with one or more inactive leafs and an rm-active branch is a branch without any inactive leafs.

A. Addition and Deletion of a Node

When a leaf node wants to join a multicast connection, the nearest switch will get a FRTT (Fixed Round-Trip Time) value, which is the sum of the fixed and propagation delays from the switch to the leaf and back, by using ABR signaling messages. The ABR signaling message contains two parameters, Trm and Nrm. The default value of Trm is 100 ms and the default value of Nrm is 32 cells. The source will generate an FRM cell if either the time has exceeded Trm or Nrm–1 cells have been sent since last time the source forwarded an FRM cell. This algorithm will use four parameters, FRTT, Trm, Nrm, and ICR (Initial Cell Rate) to calculate the timeout value for the switch with the addition of the leaf node. The units of the FRTT, Trm, and ICR parameters are microseconds, milliseconds, and cells/second, respectively. The timeout value from the switch along the downstream to the leaf is calculated as follows.

$$Timeout_d = \left\lceil \frac{\frac{FRTT}{1000}}{\min(Trm, \frac{Nrm}{ICR} \times 1000)} \right\rceil \times \beta,$$

where β is a scaling factor and is set to 2 in [9].

All branches along the path from the new leaf to the root become the rm-inactive branches. It means the timeout values of these branches have to be updated by a layered scheme. The timeout update algorithm is needed whenever the timeout value of a branch is updated and is described as follows. First, the algorithm compares the calculated timeout value with the STV and updates the STV if the timeout is greater than the STV. The timeout update algorithm stops if the timeout is not greater than the STV.

If the timeout is greater than STV, the switch measures the round trip time (RTT) between itself and the upstream switch. The algorithm uses a control message, RTT MEASUREMENT, to get the RTT value from the switch to its upstream switch and back. Assuming that the unit of the RTT parameter is microseconds, the switch then calculates a timeout value for its upstream branch as follows:

$$Timeout_u = Timeout_d + \left\lceil \frac{\frac{RTT}{1000}}{\min(Trm, \frac{Nrm}{ICR} \times 1000)} \right\rceil \times \beta.$$

Finally, the switch sends a control message, UPDATE, which includes the new timeout value, Timeout$_u$, to its upstream

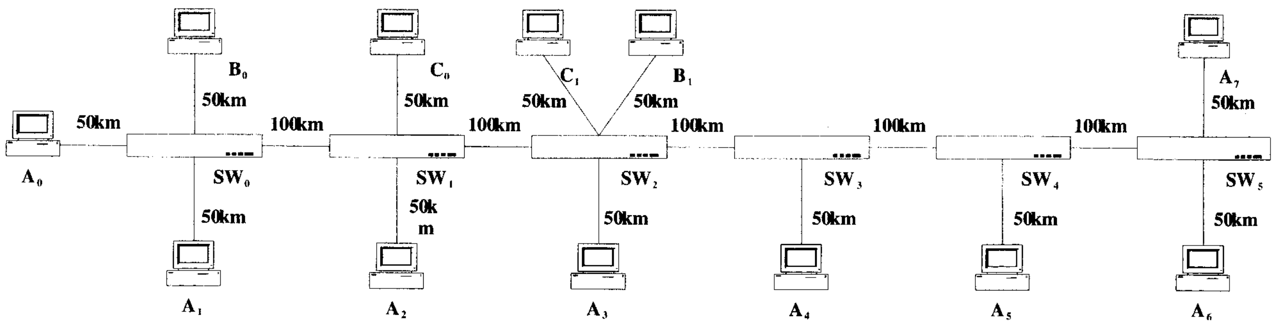


Fig. 1. A parking lot configuration.

switch. The upstream switch gets the updated timeout value of the branch and repeats the timeout update algorithm until either (1) the timeout value is less than the STV value of the switch or (2) there is no upstream node for the switch.

For example, in Fig. 1 we assume A_3 is a new node added to SW_2. Then switch SW_2 first gets the FRTT from SW_2 to A_3 and back, and uses the FRTT, Trm, Nrm, and ICR to compute a timeout value, Timeout_d. Then the computed timeout value, Timeout_d, is compared with STV. If the computed timeout value is greater than STV, the STV value of SW_2 is set to the computed timeout. Then switch SW_2 sends a control message, RTT MEASUREMENT, to upstream switch SW_1, to get RTT between switch SW_2 and switch SW_1. The RTT, Timeout_d, Trm, Nrm, and ICR are used to compute a timeout value, Timeout_u, which is included in a control message, UPDATE, and sent to switch SW_1. When switch SW_1 receives the UPDATE message, it compares the received Timeout_u with its STV. If the received timeout value, Timeout_u, is greater than its STV, the STV value of SW_1 is set to Timeout_u. The following operations in switch SW_1 are similar to switch SW_2 because it has an upstream switch, SW_0. On the other hand, if switch SW_0 receives an update message from switch SW_1, switch SW_0 only compares its STV with the received Timeout_u and changes its STV if the STV is smaller than Timeout_u since there is no upstream switch. The flowchart in Fig. 2 shows the operations of Chen's algorithm for the nearest switch and upstream switches in detail when a new node is added to a switch. It also shows that Chen's algorithm needs control messages and ABR signals to update timeout values for switches. It is complex compared to our algorithm. We will show in next section that our algorithm does not have to exchange messages between switches for computing timeout values.

Similarly, when a leaf node wants to leave a multicast connection, the leaving of the leaf node is handled recursively. When a leaf leaves a multicast tree and the nearest switch to the leaf is still in the multicast tree, the switch checks if the timeout value of the branch to the leaf is equal to the STV value. If the two values are not the same, the STV value stays unchanged. If the two values are equal, the STV should be updated as follows. When the participating branch that has the maximum timeout value is an rm-inactive branch, the algorithm first resets the STV to be zero and sets Timeout_d to be the timeout value of this branch. The timeout update algorithm is then called to update the timeout value. When the participating branch that has the maximum timeout value is an rm-active branch, the algorithm ends.

B. Adjustment of Timeout Value based on Measurements

For the rm-active branch, each switch maintains a counter, NFRM, to store the number of FRM cells received between two unmarked BRM cells. It then updates the timeout value according to the formula:

$$Timeout_t = \alpha Timeout + (1 - \alpha)(NFRM \times \beta),$$

where α ($0 < \alpha < 1$) is a smoothing factor that determines the relative weight given to the old value and β ($\beta > 1$) is a scaling factor. The value of NFRM is decreased if the number of FRM cells received between two BRM cells is reduced. From the above equation, we can see the timeout is also decreased. However, the value of NFRM may become smaller due to the decrease of bandwidth in the forward direction. The decrease of timeout value may lead to mistaking a responsive switch as a non-responsive branch. Although this can be avoided by having a larger α value (close to 1), we then have a slower response time when there is really a non-responsive branch. The effects of a larger α value are also shown in two simulations in Section IV.

III. TIMEOUT ALGORITHM

We will explain possible problems of Chen's timeout algorithm first and then we will describe our algorithm.

A. Possible Problems of CHEN's Timeout Algorithm

As we can observe in Chen's timeout algorithm, it still has certain problems such as the complexity of implementation, the signaling delay time, and the scalability with different network conditions. The switch has to make a lot of effort to maintain the timeout value for each branch, monitor the current conditions of the network, and handle signals between themselves and other switches. The algorithm also mistakes a responsive branch as a non-responsive branch when the source rate changes fast as shown later.

We propose a simplified timeout algorithm which can achieve approximately the same performance of the Chen's algorithm but have much lower complexity. When sources change rates frequently, our algorithm will avoid making wrong judgments of the responsive branch as Chen's algorithm.

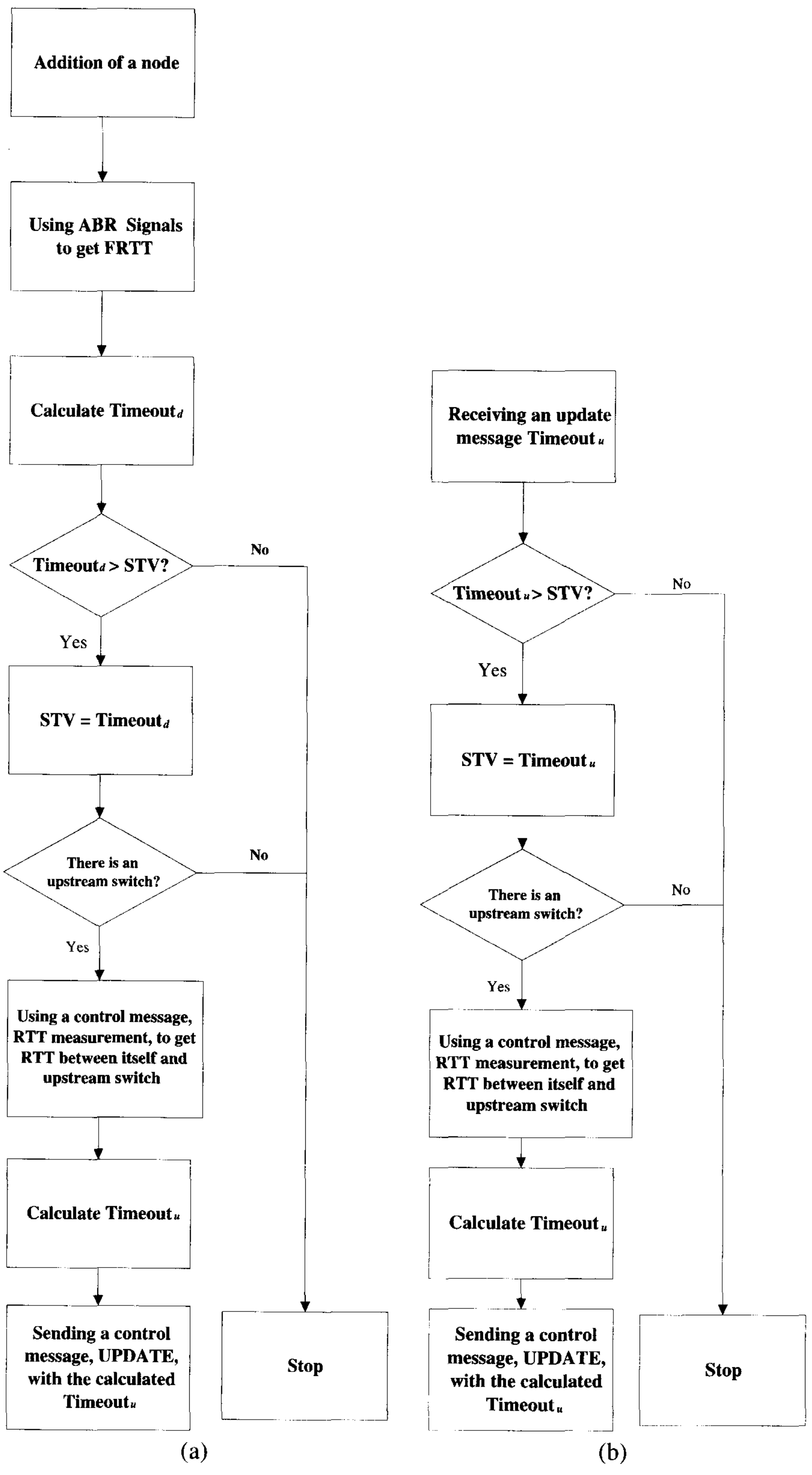

Addition of a node
Using ABR Signals to get FRTT
Calculate Timeout$_d$
Timeout$_d$ > STV?
No
Yes
STV = Timeout$_d$
There is an upstream switch?
No
Yes
Using a control message, RTT measurement, to get RTT between itself and upstream switch
Calculate Timeout$_u$
Sending a control message, UPDATE, with the calculated Timeout$_u$
Stop
(a)
Receiving an update message Timeout$_u$
Timeout$_u$> STV?
No
Yes
STV = Timeout$_u$
There is an upstream switch?
No
Yes
Using a control message, RTT measurement, to get RTT between itself and upstream switch
Calculate Timeout$_u$
Sending a control message, UPDATE, with the calculated Timeout$_u$
Stop
(b)

Fig. 2. The operations of Chen's algorithm when a new node is added to a switch: (a) the newest switches, (b) upstream switches.

B. Algorithm Design

Our timeout algorithm does not exchange messages among switches. Hence, it needs only monitor the current network condition for each branch. Instead of monitoring the number of FRM cells between two BRM cells to determine the timeout value as in Chen's algorithm, we use some common ABR parameters, which include Trm, Nrm, MCR (Minimum Cell Rate), and CCR (Current Cell Rate), to calculate the timeout value for each branch. A switch maintains a local timer for each branch connected to the switch. The switch also measures the time interval between two BRM cells. In our algorithm, we only offer an upper bound of the timeout value. The reason we use an upper bound is because the network situation changes all the time and our goal is to avoid mistaking a responsive branch as a non-responsive branch.

When a new leaf is going to join the multicast tree, the nearest switch will start to monitor the time interval between two BRM cells in this branch. We use a register, BRM_Interval, to store the value. The switch uses the following formula to calculate the timeout value for the branch (Branch_Timeout).

$$Calculatd\_Timeout = \max(\frac{Nrm}{SW[CCR]}, BRM\_Interval), \quad (1)$$

$$Branch\_Timeout = \min(Trm, Calculated\_Timeout, \frac{Nrm}{MCR}). \quad (2)$$

Here SW[CCR] is the current cell rate received from the source. BRM_Interval is the measured time value between two BRM cells received from this branch. Nrm/SW[CCR] represents a reasonable time for two interleaving FRM cell under the current value of CCR. And Nrm/MCR represents the maximum time for two interleaving FRM cells if the multicast connection has made an MCR contract during the connection setup. Note that all parameters are defined in Traffic Management 4.1. We first obtain Calculated_Timeout by choosing the maximum of Nrm/SW[CCR] and BRM_Interval. Nrm/SW[CCR] is increased when SW[CCR], the source transmission rate in forward direction, is decreased and BRM_Interval is increased when the transmission rate in backward direction is decreased. Hence, our algorithm considers transmission rates in both forward and backward directions. Then we pick the minimum of Trm, Calculated_Timeout, and Nrm/MCR as the timeout for current branch (Branch_Timeout). Trm is adopted here because Trm is the upper bound between two RM cells for an active source. In our algorithm, each branch has its own timeout value.

We apply our timeout algorithm to the "wait-for-all" algorithm. When there is no timeout, the algorithm is just the same as the "wait-for-all" algorithm. When there is timeout for a branch, the switch will ignore the feedback of the branch and send BRM cells to upstream nodes when the switch has collected all feedback from other branches. The Branch_Timeout is multiplied by the parameter β and the obtained value is used to decide whether the branch is marked as a non-responsive branch. If a branch is marked as a non-responsive branch, the switch will ignore the feedback of the non-responsive branch unless it becomes responsive again later. If it is not marked as a non-responsive branch, the Branch_Timeout value is updated by (1) and (2). In the following, CI is used to represent that there is congestion in the network, NI is used to notice the source not to increase its rate, and PCR (Peak Cell Rate) is used to regulate the maximum transmission rate of sources. Our algorithm only offers an upper bound for the timeout. The pseudo-code of our algorithm is as follows:

```
When receiving an FRM cell:
  • Multicast this FRM cell to all branches

When receiving a BRM cell: /* No timeout
If NOT BRMReceived$_i$ Then
  • Let BRMReceived$_i$ = 1
  • Let NumberOfBRMReceived = NumberOfBRMReceived + 1
  • MER = min(MER, ER from BRM cell), MCI = MCI OR CI from BRM cell, and MNI = MNI OR NI from BRM cell
  • Store the time interval between two BRM cells for branch$_i$
If NumberOfBRMReceived equals to NumberOfBranches, Then
  • Pass the BRM cell with ER = MER, CI = MCI, and NI = MNI to the source
  • MER = PCR, MCI = 0, and MNI = 0
  • Let NumberOfBRMReceived = 0
  • Let BRMReceived = 0 for all branches
Else Discard the BRM cell

When a branch has exceeded its timeout value: /* Timeout
  1. The switch will ignore the branch's feedback and send BRM cells to upstream nodes when the switch has collected all feedback from other branches
  2. If the branch has not received a BRM cell in (Branch_Timeout * $\beta$) seconds
       • Mark this branch as a non-responsive branch /* branch dead
     Else If the branch has received a BRM cell in (Branch_Timeout * $\beta$) seconds
       • Update the Branch_Timeout value according to (1) and (2)

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

When the available ABR capacity is stable, Chen's timeout algorithm can adjust the timeout value of each branch and thus reduce the time needed to handle the non-responsive branches. But when the network situation is changed fast, it has limited abilities to adjust the timeout to current network situation. We summarize the comparison of Chen's algorithm and our algorithm in Table 2.

IV. SIMULATION RESULTS

In our first simulations, we show that Chen's timeout algorithm and our timeout algorithm will both be able to detect the non-responsive branch and raise the source rate. In our second simulation, we show that Chen's algorithm will identify a responsive branch as a non-responsive branch and generate many unnecessary timeouts or have a slow response time while our algorithm will not. For both simulations, the following parameters are used:

- All links have the bandwidth of 155 Mbps.

Table 2. A comparison of two algorithms.

	Chen's algorithm	Our algorithm
Implementation complexity	Difficult to implement. Switches have to be able to communicate with each other and dynamically adjust the timeout value for each branch.	Easy to implement. Timeout value is decided locally.
Disadvantage	May mistake a responsive branch as a non-responsive branch and generate wrong timeout events.	The response time is a little longer than the response time of Chen's algorithm under few situations.

Table 3. A comparison of the starting time in handling non-responsive branch in our algorithm and Chen's algorithm.

		$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.9$
Chen's algorithm	$\beta = 1.3$	32.04 ms	32.04 ms	32.04 ms	32.32 ms	34.55 ms
Chen's algorithm	$\beta = 2$	32.04 ms	32.04 ms	32.04 ms	32.32 ms	34.83 ms
Chen's algorithm	$\beta = 3$	32.32 ms	32.32 ms	32.32 ms	32.32 ms	34.83 ms
Chen's algorithm	$\beta = 4$	32.6 ms	32.6 ms	32.6 ms	32.6 ms	35.11 ms
Our algorithm	$\beta = 1.3$	31.47 ms				
Our algorithm	$\beta = 2$	31.48 ms				
Our algorithm	$\beta = 3$	31.76 ms				
Our algorithm	$\beta = 4$	32.04 ms				

- 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 sources are persistent, i.e., there is always data to send.
- The values for ICR, MCR, and PCR of the ABR connections are set to 100 Mbps, 0 Mbps, and 155 Mbps, respectively.
- The source rate increase factor (RIF) is set to one, to allow the immediate use of the explicit rates indicated in the returning RM cells at the source.
- The parameters for Chen's algorithm are set as follows: α = 0.1, 0.3, 0.5, 0.7, and 0.9, respectively and β = 1.3, 2, 3, and 4, respectively.
- The parameter in our algorithm are set a follows: β = 1.3, 2, and 3, respectively.

A. *Two Constant CBR Connections as Background Traffic*

The parking lot configuration for the simulation is shown in Fig. 1. SW_0, SW_1, SW_2, SW_3, SW_4, and SW_5 denote six ATM switches. The configuration has one ABR multicast connection, with source A_0 and seven receivers A_1, A_2, A_3, A_4, A_5, A_6, and A_7. The configuration has another two CBR unicast connections, with a transmission rate of 10 Mbps from B_0 to B_1 and a transmission rate of 95 Mbps from C_0 to C_1. The ABR multicast connection is active from 0 ms to 110 ms and the CBR unicast connection is active from 0 ms to 110 ms. The link between SW_0 and SW_1 is non-responsive from 30 ms to 80 ms. There are three phases in this chain configuration: (1) phase 1: 0 ms to 30 ms, (2) phase 2: 30 ms to 80 ms, and (3) phase 3: 80 ms to 110 ms.

At phase 1, the ABR multicast connection shares the link between SW_1 and SW_2 with two CBR unicast connections from 0 ms to 30 ms. Hence, the source rate of the ABR traffic is decreased from 100 Mbps to 50 Mbps and maintains that rate until the network condition is changed at 30 ms.

At phase 2, the branch between SW_0 and SW_1 is non-responsive at 30 ms and active again at 80 ms. Chen's timeout algorithm will be able to find out that the branch is non-responsive and inform the source A_0 to raise the sending rate at the time between 32.04 ms to 35.11 ms, depending on different setting of parameters α and β. Our timeout algorithm will be able to find out that the branch is non-responsive and inform the source A_0 to raise the transmission source at the time between 31.47 ms to 32.04 ms, depending on different values of β. Note that our algorithm only depends on a variable β.

At phase 3, since the branch is active again, our algorithm will be able to detect that the branch is responsive and hence adjust the source rate according to the network condition. Our algorithm will decrease the source rate at around 82.27 ms. Chen's algorithm is also able to detect this situation and respond at around 82.23 ms. Note that the time is not related to α and β, and is only influenced by the transmission delays from the RM cells. We summarize the response time of detecting the timeout events in Table 3. Fig. 3 shows an example of the timeout value of the branch between SW_0 and SW_1 in Chen's algorithm. We can find that the timeout value will increase to a large value and dynamically adjust to a very small value after the branch is responsive again. The large variations of timeout values could cause wrong judgments of timeout events which will be shown in the following simulation.

B. *An On-Off CBR Background Traffic and a Constant CBR Background Traffic*

The network configuration is the same as the configuration in Fig. 1. The ABR multicast connection is active from 0 ms to 130 ms. The CBR source B_0 is still a constant CBR traffic with a transmission rate of 10 Mbps. However, the CBR source C_0

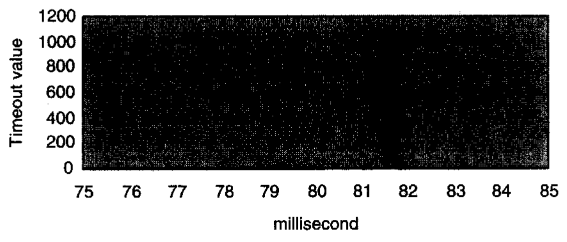


Fig. 3. Timeout value for the branch between SW_0 to SW_1 in Chen's Algorithm (α = 0.1, β = 2).

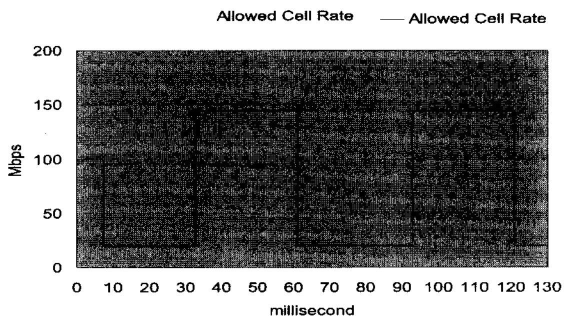


Fig. 4. Allowed cell rate for ABR source for both algorithms in simulation 2.

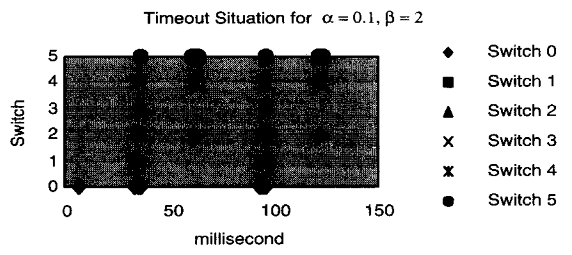


Fig. 5. Wrong timeout events detected by Chen's algorithm (α = 0.1, β = 2).

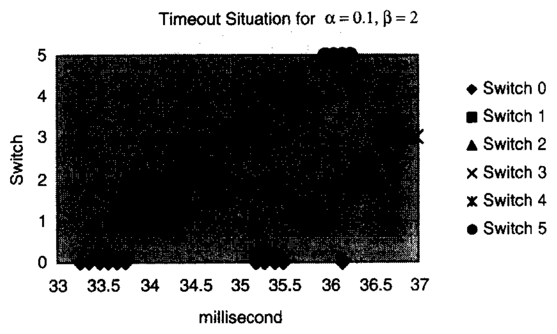


Fig. 6. Wrong timeout events between 33 ms and 37 ms detected by Chen's algorithm (α = 0.1, β = 2).

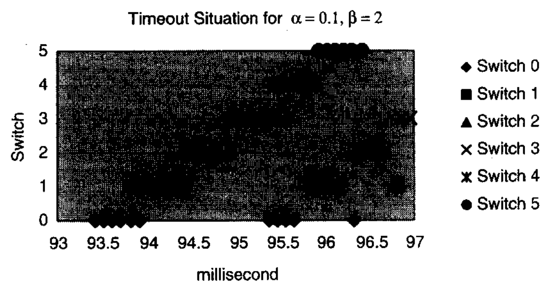


Fig. 7. . Wrong timeout events between 93 ms and 97 ms detected by Chen's algorithm (α = 0.1, β = 2).

is changed to become an on/off source and its transmission rate is 125 Mbps. The "on" period and "off" period are interleaved with 30 ms each. In this simulation, the allowed cell rate of the ABR traffic is limited by the CBR source. When the CBR source is on, the ABR source rate is limited to 20 Mbps. When the CBR source is off, the ABR source sending rate can be increased to 145 Mbps.

The simulation results are shown from Fig. 4 to Fig. 7. Fig. 4 shows the allowed cell rate of ABR source for both algorithms. Because the allowed cell rate is increased from 20 Mbps to 155 Mbps at around 30 ms and 90 ms, there are timeouts in Chen's algorithm as shown in the following figures. Fig. 5 shows the wrong timeout events detected by Chen's algorithm in switch 0, 1, 2, 3, 4, and 5 with α=0.1 and β=2. Because we see there are many timeouts at around 33 ms to 37 ms, and around 93 ms to 97 ms, we draw these periods in detail. Fig. 6 shows the timeout events have "propagation effects" from switch 0 to switch 5. Fig. 7 also shows the "propagation effects". As we can see in the simulation results, Chen's timeout algorithm will misjudge a responsive branch and generate many timeouts when the source rate exhibits oscillations. Our timeout algorithm will not suffer this problem. The number of mistaken timeouts is shown in Table 4. The reason is that because the source is changed rapidly, it is very hard to use the FRM cells as a basis to decide whether the branch is non-responsive. It happens very often that many FRM cells have passed the switch while receiving very few BRM cells. Under that condition, there will be timeouts detected in Chen's algorithm when α is not large. However, in fact there is no non-responsive branch and the detection of timeout events is wrong, which will result in incorrect calculation of available bandwidth and may lead to congestion. When α is as large as 0.9, although there are no timeouts detected in simulation 2, we can find that the response time is much slower when there is really a non-responsive branch as shown in Simulation 1. In our algorithm, we consider both the arrival rates of FRM cells and BRM cells. Hence, the wrong detection of timeouts is properly avoided in the simulation.

V. CONCLUSION

Chen proposed a timeout algorithm to handle the "non-responsive" branches. The main problem of Chen's algorithm is its over complexity. It requires the switches to update the timeout values recursively and demands switches and the leaf nodes to exchange signals. We proposed a simple timeout algorithm which can handle the non-responsive branches and is suitable for the situation when the network condition changes dynamically. Our algorithm will not require signals between switches and between leaf nodes and switches. Hence, the complexity of

Table 4. A comparison of the number of timeouts detected for the two algorithms.

		$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.9$
Chen's algorithm	$\beta = 1.3$	171	170	136	106	0
Chen's algorithm	$\beta = 2$	158	147	132	92	0
Chen's algorithm	$\beta = 3$	130	102	94	72	0
Chen's algorithm	$\beta = 4$	97	68	56	48	0
Our algorithm	$\beta = 1.3$	0				
Our algorithm	$\beta = 2$	0				
Our algorithm	$\beta = 3$	0				
Our algorithm	$\beta = 4$	0				

our algorithm is much lower. Simulation results showed that our timeout algorithm can handle the non-responsive branches and utilize the available bandwidth within a small period of time. Especially, when the network conditions are changed fast, Chen's algorithm may not be able to respond to them correctly and mistake a responsive branch as a non-responsive branch while our algorithm could avoid making mistaken assessments of timeout events.

ACKNOWLEDGEMENT

This research was supported in part by NSC of Taiwan under contract 88-2213-E-110-021.

REFERENCES

[1] Y.-Z. Cho and M.-Y. Lee, "An efficient rate-based algorithm for point-to-multipoint ABR service," in *Proc. IEEE GLOBECOM'97*, Nov. 1997.

[2] *Traffic Management Specification 4.0*, The ATM Forum Technique Committee, Apr. 1996.

[3] L. Roberts, "Rate based algorithm for point-to-multipoint ABR service," in *ATM Forum Contribution* 940772, 1994.

[4] H. Y. Tzeng and K. Y. Siu, "On max-min fair congestion control for multicast ABR service in ATM," *IEEE J. Select. Areas Commun.*, vol. 15, no. 3, Apr. 1997.

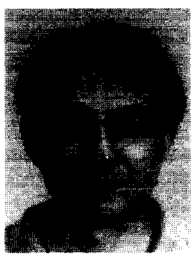
[5] W. Ren, K. Siu, and H. Suzuki, "On the performance of congestion control algorithm for multicast ABR service in ATM," in *Proc. IEEE ATM'96*, Aug. 1996.

[6] S. Fahmy *et al.*, "Feedback consolidation algorithms for ABR point-to-multipoint connections in ATM networks," in *Proc. IEEE INFOCOM'98*, 17th Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings, IEEE, vol. 3, 1998, pp. 1004–1013.

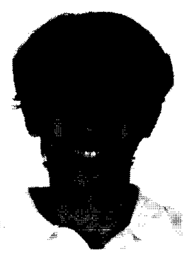
[7] T. Jiang, E. W. Zegura, and M. Ammar, "Improved consolidation algorithms for point-to-multipoint ABR service," in *Proc. IEEE ATM Workshop'98*, 1998, pp. 195–201.

[8] D.-H. Kim *et al.*, "A scalable consolidation algorithm for point-to-multipoint ABR flow control in ATM networks," in *Proc. IEEE ICC'99*, vol. 1, 1999, pp. 118–123.

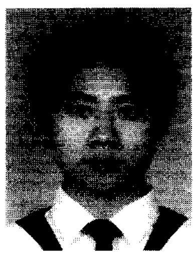
[9] H.-S. A. Chen and K. Nahrstedt, "Feedback consolidation and timeout algorithms for point-to-multipoint ABR service," in *Proc. IEEE ICC'99*, vol. 1, 1999, pp. 135–139.



Wei Kuang Lai received the BS degree in Electrical Engineering from National Taiwan University in 1984 and the Ph.D. degree in Electrical Engineering from Purdue University in 1992. He joined the faculty of Department of Computer Science and Engineering, National Sun Yat-Sen University in August 1992 and is now an associate professor. His research interests are in high-speed networks and wireless networks.



Chien Ting Chen received the M. S. degree in Department of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan, in 2000. His main interest is in IP flow controls over ATM networks.



Chilin Li received the M. S. degree in Department of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan, in 2003. His main interest is in network protocols and mobile ad hoc networks.