IMPROVING THE ROBUSTNESS OF FUZZY LOGIC ATM ABR RATE CONTROL FOR LARGE ROUND TRIP TIMES

Lachlan L. H. ANDREW and KUAN Su-Hsien
Department of Electrical and Electronic Engineering
University of Melbourne, Parkville, Vic, 3052 Australia
L.Andrew@ee.mu.oz.au +613 9344 9201 (ph) +613 9344 9188 (fax)

Abstract

For ABR to be a practical part of ATM, its rate must be able to be controlled effectively even in the presence of a large round trip time (RTT). A simple fuzzy logic controller was recently shown to perform poorly in networks with large RTTs. This paper compares three methods of improving the robustness of this controller to large RTT. One of these in particular is found to provide significantly lower cell loss for large RTTs, and higher utilisation for very large RTTs.

1 Introduction

ATM aims to combine both real time and non-real time connections in a single broadband network. In order to achieve high utilisation, ATM networks must provide an available bit rate (ABR) service [1–3], which allows very bursty, non-realtime sources to transmit at high rates when network load is low, and low rates when either network load is high, or their own demand is low. The key to ABR services is rate control [3]. Like all ATM connections, ABR data is divided into 424 bit cells, and the data rate is controlled by adjusting the cell rate. Rate control involves providing each source with information about its allowed cell rate (ACR) in time for it to adjust its transmission rate. To this end, ABR sources periodically transmit resource management (RM) cells, which follow the same path as the data cells, and then return to the source. One of the fields in an RM cell is an explicit rate (ER), which can be set by a congested switch to provide an upper bound on the ACR of the source. Each switch the RM cell passes through may reduce the ER, but may not increase it. ABR rate control usually requires each switch to determine the appropriate ER value to indicate.

In recent years, fuzzy logic [4,5] has shown promise in the control of complex systems, and several fuzzy controllers have been proposed for ABR rate control [6,7]. Fuzzy explicit rate marking (FERM) presented in [6] was recently shown to perform poorly in the presence of large round trip times (RTTs) [8]. This paper will compare several techniques for improving the performance of FERM, both within the fuzzy logic framework, and using a hybrid fuzzy/classical approach.

A brief review of fuzzy logic concepts, and a description of the original FERM algorithm, will be given in Section 2, followed by a description of the modified algorithms in Section 3. Section 4 will present and discuss the findings of this investigation. The specific fuzzy rules of FERM are described in the appendix.

2 Description of FERM

2.1 Overview of Fuzzy Logic

Experts rely on vague, imprecise rules to make decisions. Fuzzy logic [4,5] is an attempt to formalise the processing of such imprecise rules. Importantly, it allows a variable to be partly “large” and partly “small” at the same time. Words like “large” and “small” are called *linguistic terms*, and a fuzzy value is essentially a measure of how well each of a collection of linguistic terms describe a particular quantity. How well the term describes the value is called the membership of the fuzzy value in the fuzzy set. For example an ACR of 300 cells/ms may be “0.5 large and 0.3 very large”. Application of fuzzy logic has three steps: fuzzification (conversion of measured values into fuzzy values), rule application, and defuzzification (the reverse of fuzzification).

Fuzzy rules have the form “IF *premise* THEN *conclusion*”. The premise is made up of statements “*variable* IS *value*”, combined with the operators AND, OR and NOT. The truth of a statement “*variable* IS *value*” is simply the membership of the fuzzy value of “variable” in the fuzzy set corresponding to the linguistic term “value”. The truth of “*exp*1 AND *exp*2” is the minimum of *exp*1 and *exp*2, and the truth of “*exp*1 OR *exp*2” is the maximum of *exp*1 and *exp*2.

2.2 The FERM algorithm

FERM [6] is an *explicit rate* rate control algorithm, which means that it calculates the desired maximum transmission rate of each source and explicitly specifies this rate through the mechanism of resource man-

agement (RM) cells, which are generated by the source every N_{rm} cell times. Time is divided into control intervals, each consisting of N_{fp} cell arrivals, and the state of the controller is updated every control interval. To update the controller state, the controller applies the fuzzy rules given in the appendix to the current buffer occupancy, q, and the difference, dq, between q and the buffer occupancy at the previous control time. The algorithm then produces a desired fractional flow rate (FFR), which is multiplied by the peak cell rate (PCR) of each ABR connection to determine the explicit rate (ER) for that connection at this switch. That is, the algorithm sets

$$ER = F(q, dq), \tag{1}$$

where F is the connection's PCR times the FFR given by the fuzzy rules, and ER is the value placed in the explicit rate field of the RM cell. For each RM cell arriving in that control interval, if the switch's ER for the corresponding connection is lower than the ER already in the RM cell, then the old ER value is overwritten.

3 Robust FERM

This paper will consider three very simple modifications to the original FERM approach, and compare their effectiveness at improving the robustness of the system to large round trip times. All of these modification make the control more conservative and thus improve the cell loss performance at the expense of the utilisation.

The first two modifications are both based on the standard notion that caution is required when increasing the rate, and swift action is required when reducing it. That is the principle behind the "additive increase, multiplicative decrease" rule often used in flow control. However, the original FERM provides an absolute value for the new ER, independent of the current value. Thus there is no concept of "increasing" or "decreasing" the rate, but only of the "correct" rate. This can be rectified in two ways. The first way considered here is to introduce a third input variable to FERM indicating the current ACR, which is indicated in the incoming RM cell. The control rule thus becomes

$$ER = F_{3var}(q, dq, ACR), \tag{2}$$

where F_{3var} is the connection's PCR multiplied by the FFR determined by a *modified* set of fuzzy rules. The rules used are presented in Table 4, although other rules are also possible. These rules indicate that the fuzzy rate should not increase by more than one "level" for a single RM cell arrival, no matter how lightly loaded the system currently is. If the network is congested, the rate must still be reduced sufficiently quickly, and the original FERM rules apply.

Another way to enforce the slow increase would be to step out of the fuzzy paradigm and use a hybrid fuzzy/classical approach. In this approach, the advised explicit rate from FERM would be used as the input for a simple non-fuzzy formula to determine the new ACR. In this study, the rule used is

$$ER = \begin{cases} F(q, dq) & \text{if } F(q, dq) < A \\ A + \eta(F(q, dq) - A) & \text{otherwise} \end{cases}$$

where ER, $F(q, dq)$, q and dq are the same as for (1), A is the current ACR, and η is a step length parameter which determines the maximum rate of increase of the ACR. This approach, using the fuzzy logic directly to decrease the rate but using autoregressive (AR) increase, will be termed the "ARI" algorithm.

The third modification is simply to modify the fuzzification rules to lower the "target" queue length, as indicated in Table 3. This will force the switch to reduce the ACR sooner and thus allow more time for the source to reduce its rate in time to avoid buffer overflow. Thus longer feedback delays can be accommodated. The penalty for keeping a lower average buffer occupancy is an increased probability of emptying the buffer entirely. When the buffer is empty, the link utilisation drops below 100%, which is undesirable but still more acceptable than losing cells. This approach will be called the "low-aim" algorithm.

Because of the frequency with which control decisions must be made by ATM switches, it is important to minimise the computational complexity of the rate control algorithm. The low-aim scheme has the same complexity as the original FERM algorithm. Both of the other algorithms involve an increase in computation. The ARI algorithm involves only minimal extra computation: three floating point additions and one multiplication for each decision. However, the 3var algorithm entails a substantial increase in computation, involving the processing of twice as many fuzzy rules, most of which involve twice as many fuzzy operations as the original FERM algorithm.

4 Results

4.1 Experimental scenario

The network investigated in this paper consists of one ABR source and one VBR source attached to a single switch with a 1024 cell buffer and a bottleneck output link of 155 Mbps. The ABR sources were persistant sources (i.e., they always have data to send) with a PCR equal to the maximum cell rate of the bottleneck link to model a file transfer. The VBR source was a slow-start on-off source with a PCR of one quarter of the rate of the bottleneck link. That is, when the source turns on, its bit rate ramps up linearly to one quarter of the link rate and then is constant until it turns off. The parameters are given in Table 1. The switch periodically checks each of the sources, and deems the number of currently active sources the be the number which have transmitted cells since the last check.

Table 1: Simulation parameters for ABR sources. (Note: 365 cells/ms corresponds to 155 Mbps)

Name	Description	Value
MCR	Minimum cell rate	0.365 cells/ms
PCR	Peak cell rate	365 cells/ms
ICR	Initial cell rate	365 cells/ms
N_{rm}	Cells per RM cell	10
N_{fp}	Cells per control interval	50

The four control algorithms were tested for the case of a single ABR source starting transmitting at time 0 ms, with a VBR source starting at time 3 ms and reaching full rate by 33 ms, for a range of round trip times. Note that the ABR control algorithm has no control over the rate of the VBR source. When the VBR source starts transmitting, there will be transients while the controller determines the appropriate ABR rate, generally followed by sustained oscillatory behaviour due to the large RTTs under investigation. The results here are quoted for the "steady state" oscillations, after the initial transients. The buffer will often overflow during the initial transients, because the ABR source must keep transmitting at its PCR for at least one RTT until the first indication of congestion returns from the network. Such overflow can only be avoided by forcing VBR and CBR sources to increase their rates very slowly, and will thus be ignored in this study.

For this study, the step length parameter of ARI was $\eta = 0.8$.

4.2 Simulation results

The two most important performance measures for ABR are the link utilisation, and the probability of cell loss, since ABR connections are inherently insensitive to delay. Figure 1 shows the steady state link utilisation as a function of round trip times for each of the four control algorithms. The solid lines correspond to acceptable steady states, in which the buffer does not fill completely, and the dashed lines correspond to unacceptable steady states. For low delays, all of the algorithms have 100% utilisation. As the delay increases above 2 ms, the utilisation drops for both ARI and low-aim. When the round trip time reaches 4 ms, the utilisation also drops for the original algorithm original and 3var. The three pure fuzzy-logic schemes clearly trade utilisation for stability, with the more stable schemes consistently providing lower utilisation.

Much better results than for any of the pure fuzzy logic schemes were obtained by the hybrid scheme. Despite its lower utilisation for moderate RTTs, ARI suffers a much more gradual drop in utilisation as the delay gets very large, and is better than the original scheme for RTTs over 6 ms. However, the real benefit of the hybrid scheme is that it produces no cell loss over the entire range of RTTs studied in this simulation. This is shown clearly in Figure 2 which shows the

Figure 1: Utilisation vs round trip time for the original FERM and the three enhanced techniques.

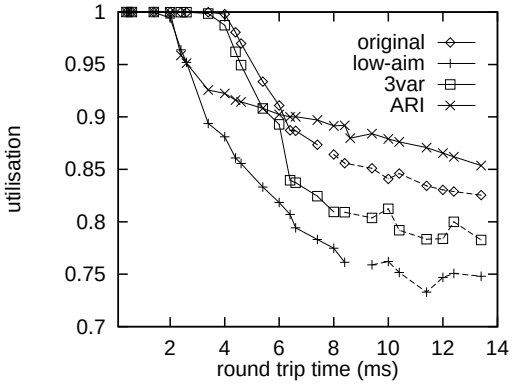
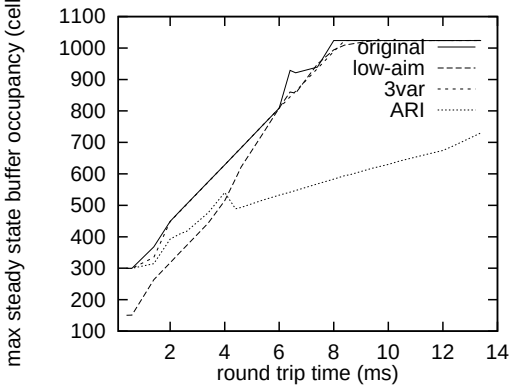


Figure 2: Maximum buffer occupancy in steady state for the original FERM and the three enhanced techniques.



maximum buffer occupancy once the scheme reaches steady state. For low RTTs, the low-aim algorithm successfully keeps the maximum buffer occupancy very low. However, as the RTT increases, its performance degrades and the maximum occupancy approaches that of the original and 3var algorithms, and the buffer overflows for a similar value of RTT. In contrast, the hybrid technique has a higher occupancy for low RTTs, but maintains a much lower occupancy as the RTT increases. Extrapolating from the graph indicates that it will not cause cell loss for RTTs as high as 20 ms, compared to 8 ms for the pure fuzzy logic techniques.

The ARI algorithm can be tuned by altering η. Preliminary results show that with $\eta = 0.2$, the utilisation is almost 100% until RTT=4 ms, after which it drops, but remains above that of the original algorithm. Cell loss occurs at RTT=10 ms, when the utilisation is 85%.

The actual temporal behaviour of the system can be seen for RTT=9 ms in Figures to 8. Note that the initial response is essentially the same for all schemes because of the time taken for the source to receive notification of the congestion. Note also that the queue length using ARI is 0, indicating suboptimal utilisation, more often than for the other schemes (Figure 4). However,

Figure 3: Queue length vs time for 3var and original FERM.

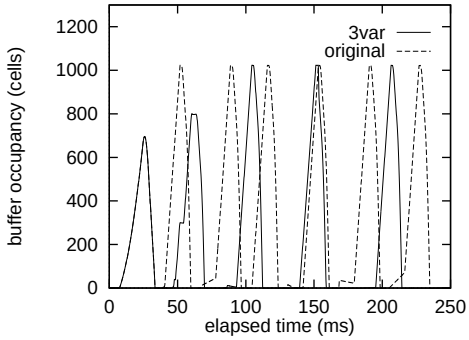
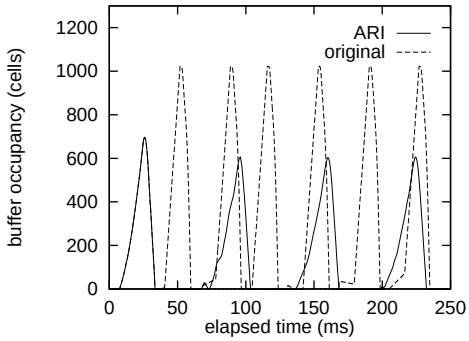


Figure 4: Queue length vs time for ARI and original FERM.



due to the reduced amplitude of the oscillations in the ACR, the actual ACR at these times never reaches zero (Figure 7), and so the resulting utilisation is still higher.

5 Conclusion

Recent results [8] indicated that one proposed fuzzy controller, FERM, performs poorly in the presence of large round trip times. The results presented here indicate that very simple modifications to FERM can provide improvements in robustness in this case. In particular, a hybrid fuzzy/classical algorithm consistently provides substantially lower cell loss, and for large round trip times also provides greater utilisation of the network. This comes at negligible additional computational complexity. This modification increases the permissible round trip time from about 8 ms for FERM to about 20 ms.

Appendix

A linguistic values in FERM is defined by four values (x_1, x_2, x_3, x_4). For example, the linguistic value `moderate` is defined by (220, 380, 550, 820). An input value less than 220 or greater than 820 has zero

Figure 5: Queue length vs time for low-aim and original FERM.

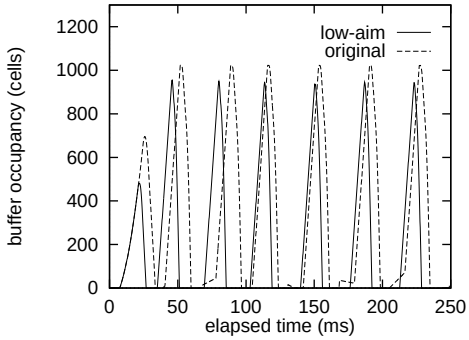
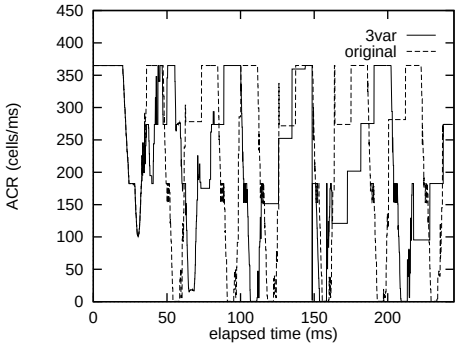


Figure 6: Allowed cell rate (ACR) vs time for 3var,



membership in the set moderate, and an input value in the range (380, 550) has unity membership in the set, so that the truth value of the statement "`400 is moderate`" is 1. The membership increases linearly between x_1 and x_2, so that "`300 is moderate`" has a truth value of 0.5. FERM uses three linguistic variables, `q` (queue length), `dq` (change in queue length) and `rate` (allowed cell rate divided by the PCR, in the range -0.2 to 1.2). The possible linguistic values for each of these are given in Table 2.

The modified fuzzification rules for `q` in the low-aim algorithm are given in Table 3.

The linguistic rules defined in terms of these variable and values are given in Table 4. The rules in bold, with only `q` and `dq` as inputs, are the original FERM rules, and the entire set of rules with all three inputs is the modified set of rules for scheme 3var. The linguistic variable `ACR` takes on the same values as the variable `rate`.

References

[1] T.M. Chen, S. S. Liu and V. K. Samalam, "The available bit rate service for data in ATM networks", *IEEE Comms. Mag.*, pp. 60–71, May 1996.

Figure 7: Allowed cell rate (ACR) vs time for ARI.

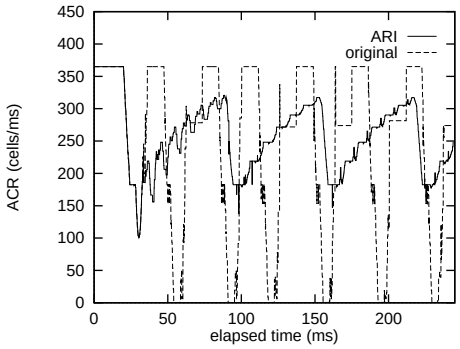
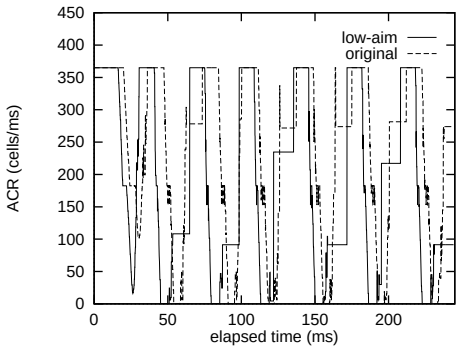


Figure 8: Allowed cell rate (ACR) vs time for low-aim.



[2] The ATM Forum, *Traffic Management Specification Version 4.0*, April 1996.

[3] R. Jain *et al.*, "Source behaviour of ATM ABR traffic management: An explanation", *IEEE Comms. Mag.*, pp. 50–57, Nov., 1996.

[4] M. Jamshidi, N. Vadiee and T. J. Ross (eds), *Fuzzy logic and control: software and hardware applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[5] R. R. Yager and D. P. Filev, *Essentials of fuzzy modeling and control*, John Wiley, New York, 1994.

[6] A. Pitsillides, Y. A. Sekercioglu and G. Ramamurthy, "Effective control of traffic flow in ATM networks using fuzzy explicit rate marking (FERM)", *IEEE J. Select. Areas. Commun*, vol. 15, no. 2, pp 209–225, February 1997.

[7] V. Catania, G. Ficili, S. Palazzo and D. Pano, "A comparative analysis of fuzzy versus conventional policing mechanism for ATM networks", *IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp. 449–459, June 1996.

[8] S.-H. Kuan and L. Andrew, "Performance of Fuzzy Logic ABR Rate Control with Large Round Trip Times", to appear in *Proc. Globecom 98*.

values for q				
empty	0	0	200	400
moderate	220	380	580	820
full	600	830	1024	1024
values for dq				
down_fast	-500	-500	-200	-85
down_slow	-158	-78	-78	-12.5
zero	-60	0	0	60
up_slow	12.5	78	78	158
up_fast	85	200	500	500
values for rate				
v_little	-0.20	0.00	0.00	0.20
little	0.5	0.25	0.25	0.45
moderate	0.30	0.50	0.50	0.70
high	0.55	0.75	0.75	0.95
v_high	0.80	1.00	1.00	1.20

Table 2: Linguistic variables and their possible linguistic values.

values for q				
empty	0	0	100	200
moderate	100	200	300	500
full	300	500	1024	1024

Table 3: Modified linguistic values for buffer occupancy, q.

if q is	and dq is	and ACR is	then rate is
empty		v_high or high	**v_high**
empty		moderate	high
empty		little	moderate
empty		v_little	little
moderate	**down_fast**	v_high or high	**v_high**
moderate	down_fast	moderate	high
moderate	down_fast	little	moderate
moderate	down_fast	v_little	little
moderate	**down_slow**	v_high or high	**high**
moderate	down_slow	moderate	high
moderate	down_slow	little	moderate
moderate	down_slow	v_little	little
moderate	**zero**	not v_little	**moderate**
moderate	zero	v_little	little
moderate	**up_slow**		**little**
moderate	**up_fast**		**v_little**
full	**down_fast**	not v_little	**moderate**
full	down_fast	v_little	little
full	**down_slow**		**little**
full	**zero**		**v_little**
full	**up_slow**		**v_little**
full	**up_fast**		**v_little**

Table 4: Linguistic rules for three-variable FERM. The rules in bold are the original two-variable rules.