Performance of Fuzzy Logic ABR Rate Control with Large Round Trip Times

KUAN Su-Hsien and Lachlan L. H. ANDREW

School of Dental Science — Department of Electrical and Electronic Engineering

University of Melbourne, Parkville, Vic 3052, Australia

Abstract—**For ABR to be practical, its rate must be able to be controlled effectively even in the presence of a large round trip time (RTT). This paper compares the performance of a recently proposed fuzzy logic controller for ABR with a simple threshold based scheme in this case. The fuzzy logic scheme is found to be significantly more sensitive to increases in the RTT than the non-fuzzy scheme.**

I. Introduction

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], that is, providing each source with information about its allowed cell rate (ACR) in time for it to adjust its transmission rate. However, uncertainties in transmission delays and traffic patterns make this a very complex task.

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]. This paper will compare the performance of a recently proposed fuzzy explicit rate mechanism for ABR control, FERM [6], with a conventional controller based on crisp (non-fuzzy) rules [8,9]. The comparison will focus on the performance of these schemes in the presence of large round trip times (RTTs), which can cause instability in control algorithms.

A brief review of fuzzy logic concepts, and a description of the FERM algorithm, will be given in Section II, followed by a description of the classical RRY algorithm in Section III. Section IV will present and discuss the findings of this investigation. The specific fuzzy rules of FERM are described in the appendix.

II. Description of FERM

A. Overview of Fuzzy Logic

Experts rely on vague, imprecise rules to make decisions. They consider whether an object is "large" or "small", rather than whether its size exceeds a precise threshold. Fuzzy logic [4,5] is an attempt to formalise the processing of such imprecise rules. 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. Several linguistic terms may describe a value, but none are likely to describe it perfectly. 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". (Note that the memberships must be in the range (0, 1), but need not sum to 1). Engineering applications of fuzzy logic have three steps: fuzzification, rule application, and defuzzification. In fuzzification, crisp input values measured from the system, like the current ACR, are converted to fuzzy values. Defuzzification is the reverse of this operation, in which the degree of membership in each of the sets is considered, and a single crisp value is produced.

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; for example "buffer capacity IS high AND rate of change IS very low". 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.

B. 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 management (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 occu-

The authors were with the Department of Communication and Electronic Engineering, RMIT, 124 Latrobe St, Melbourne, Vic 3000, Australia pancy 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. 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.

III. Description of RRY

The benchmark to which FERM is compared in this paper is an algorithm based on work of Ramamurthy, Ren and Yin [8,9] which we will call the RRY algorithm. This is based on the approach of allocating each link a fair share of the available bandwidth, which is the bandwidth once the sum of the minimum cell rates (MCRs) of the sources has been removed. The fair share, FS, is then the available bandwidth divided by the number of ABR sources sharing it, which is assumed to be known. Ideally, all sources should transmit at exactly their fair share rate all of the time. However, to ensure stability, operating conditions are divided into periods of congestion and no congestion, depending on the buffer occupancy. In this study, the switch was deemed to be congested if the buffer was more than half full. During congestion, the explicit rate is slightly lower than the fair share:

$$ER = MCR + (RDF \times FS)$$

where RDF is a rate decrease factor, chosen as 0.875 here. Similarly, when there is no congestion, the explicit rate is slightly higher than the fair share by an amount proportional to the peak cell rate of the connection:

$$ER = MCR + FS + (PCR \times RIF)$$

where RIF is the rate increase factor, chosen as 0.003 here.

IV. Results

A. Experimental scenario

The network investigated in this paper consists of N_{src} ABR sources 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 with a PCR equal to the maximum cell rate of the bottleneck link to model a file transfer, and the VBR source was an on-off source with a PCR of one quarter of the rate of the bottleneck link. The parameters are given in Table I. 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.

B. Single ABR Source

The two 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, for a range of round trip times. Note that the ABR control algorithm has no control over the rate of the VBR source. Figure 1 shows the buffer occupancy against time for some of the significant values of RTT using the fuzzy control algorithm. For low delays, the systems behaves qualitatively like a simple damped second order system. For an RTT of above approximately 1.43 ms, the system is no longer damped, but shows sustained oscillation, whose magnitude increases as the RTT increases. For increasing RTT, oscillations also become less sinusoidal, and begin to resemble triangular waves with rounded peaks and troughs. For a RTT of less than 4.00 ms, the buffer is never empty, and so the utilisation is 100% (neglecting overheads such as RM cells). However, for larger RTTs, the buffer is empty for a proportion of each cycle, and the utilisation drops off steadily. When the RTT reaches 8.47 ms, the oscillations become large enough to overflow the buffer, causing unacceptable cell loss. For these large delays, the system is very clearly not a simple second order system, with the piece-wise nature of the fuzzy rules becoming apparent for a queue size of around 700 cells and decreasing. Delays this large would not be expected in such a simple single switch system, since they correspond to the propagation delay of a 2500 km link, but such delays are commonplace in systems containing multiple switches.

Much better results were obtained using the classical RRY algorithm, as can be seen in Figure 2. Under this

TABLE I
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
RIF	Rate increase factor	0.003
RDF	Rate decrease factor	0.875
N_{rm}	Cells per RM cell	10
N_{fp}	Cells per control interval	50

Fig. 1. Buffer occupancy vs time for several values of RTT for 1 ABR source and one VBR source starting at 3ms with FERM control.

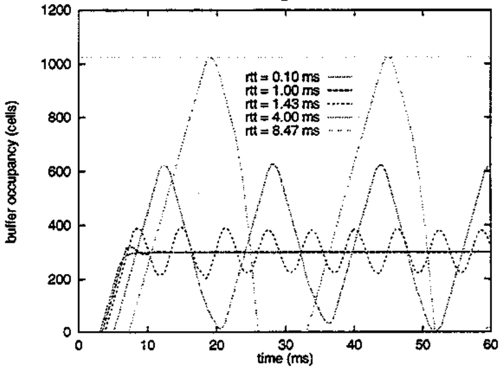


Fig. 2. Buffer occupancy vs time for several values of RTT for 1 ABR source and one VBR source starting at 3ms with RRY control.

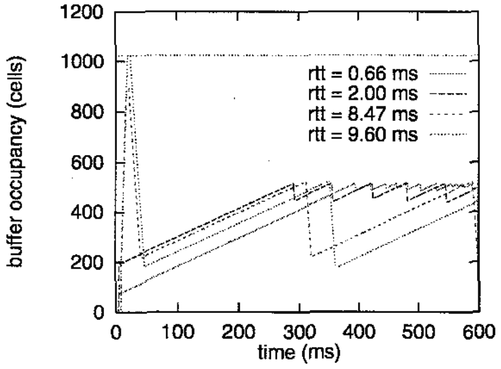


Fig. 3. Total source rate and queue length vs time. RTT = 0.17 ms. FERM control

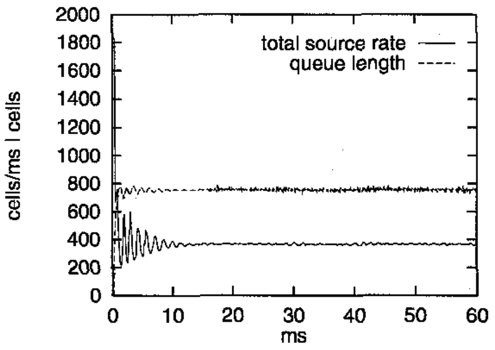
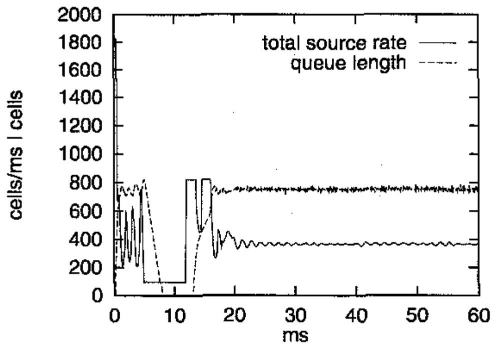


Fig. 4. Total source rate and queue length vs time. RTT = 0.18 ms. FERM control



scheme, a backlog immediately builds up when the VBR source starts transmission, but after one RTT, the control algorithm cuts the rate and the system remains relatively stationary. As the RTT increases, the initial backlog increases. Cells will be lost when the size of this backlog exceeds the buffer size, which in this simulation occurs for RTT = 9.6 ms, which is greater than the RTT causing loss for the FERM algorithm. The typical behaviour can be clearly seen from the graph for RTT = 8.47 ms. Initially the buffer fills rapidly, and once the control has commenced the buffer occupancy will continue to rise gradually until one RTT after the threshold has been exceeded, at which time it will drop rapidly until one RTT after the occupancy drops below the threshold again. Thus, unlike the results of the FERM algorithm, there will be sustained low amplitude oscillations even for low RTTs. However, moderate oscillations are not of themselves harmful; it is much more important to ensure that the utilisation of the bottleneck link is 100% and the cell loss ratio (CLR) is zero. Thus oscillations which cause the buffer to empty entirely or cause the buffer to fill entirely are the primary concern. Since these phenomena occur at significantly larger levels of RTT for the RRY algorithm than FERM, the RRY algorithm is superior in the presence of large delays.

C. *Multiple ABR sources*

The results presented above show how a single ABR source will react to a step change in available bandwidth. However, in a practical situation, there are likely to be many ABR sources all interacting with one another. When one source reduces its rate, other sources will perceive a reduction in the rate of increase of the queue and thus increase their rates.

To investigate the scalability of the FERM and RRY algorithms, simulations were performed for RTTs of 0.17 to 0.2 ms with five ABR sources starting at approximately time zero, and the same VBR source as was used in the previous section. The total source rate (the sum of all of the ABR rates and the VBR rate) and the buffer occupancy for each of these are shown in Figures 3 to 8. The RRY algorithm is essentially unaffected by this small increase in RTT. In each case, the total incoming bitrate drops rapidly to the bottleneck bitrate, and then exhibits small fluctuations about this value. The queue length quickly rises to around 500 cells and remains stable at that value. In contrast, the FERM algorithm degrades significantly as the RTT is increased. For RTT = 0.17 ms, it again approximates a damped second order response; the total source rate settles down to an almost constant bit rate after slightly over 10 ms, and the queue length remains almost constant at 800 cells. However, when the RTT is raised slightly to 0.18 ms, the response is highly non-linear. From about 6 to 12 ms, the ABR source rates all go to zero, leaving only the VBR source. The rates then settle after approximately 20 ms. When the RTT is raised to 0.2 ms the erratic behaviour lasts for 40 ms, after which the rate shows significant but decaying oscillations for well over 20 ms.

This indicates once again that fuzzy logic provides improved performance for the comparatively simple case, but shows less robustness to significant delay or a significant number of sources.

Fig. 5. Total source rate and queue length vs time. RTT = 0.2 ms. FERM control

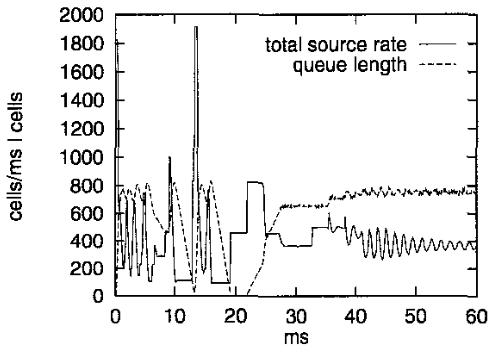


Fig. 6. Total source rate and queue length vs time. RTT = 0.17 ms. RRY control

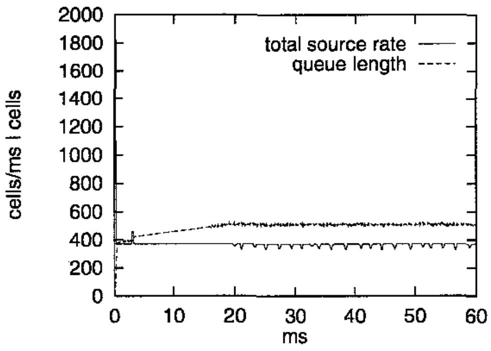


Fig. 7. Total source rate and queue length vs time. RTT = 0.18 ms. RRY control

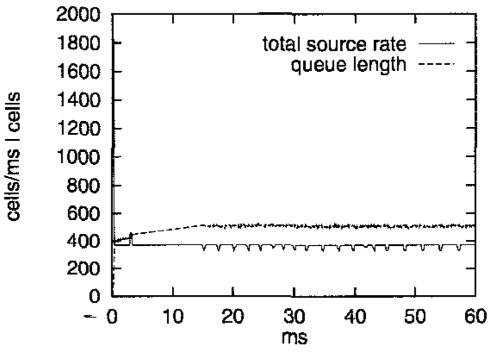
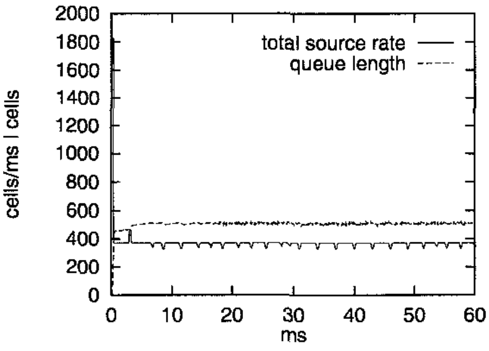


Fig. 8. Total source rate and queue length vs time. RTT = 0.2 ms. RRY control



D. *Trading sources for RTT*

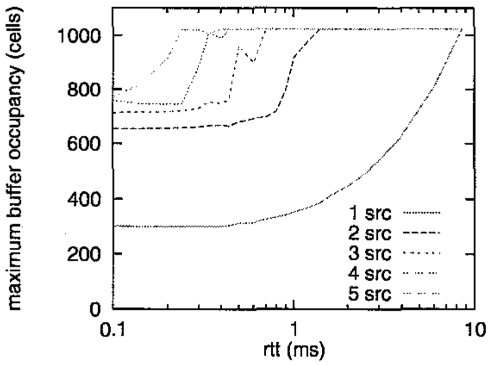
It was shown in the foregoing sections that with a single ABR source, RTT delays of the order of tens of milliseconds cause oscillations in the buffer which are sufficient to cause cell loss. It was also shown that with five mutliplexed ABR sources, erratic behaviour can be observed for much smaller RTTs. In this section, we investigate the tradeoff between the number of multiplexed ABR sources and the magnitude of RTT the system can accomodate. An RTT will be considered not to be accomodated if the buffer occupancy peaks at 100% after the initial transient period. Figure 9 shows the peak buffer occupancy as a function of the RTT for one to five sources using the FERM algorithm. For negligible delays, the peak buffer occupancy increases slowly as the number of sources increases. A more serious problem is that the rate at which the peak occupancy increases with increasing RTT is very much larger for a larger number of sources. It can be seen that the maximum possible RTT drops by an order of magnitude going from a single ABR source to five sources. This indicates that FERM does not scale well in either number of sources or network delay.

V. Conclusion

Initial results [6] indicated that a simple fuzzy controller, FERM, is an effective way to control ATM ABR services. Results presented here support this conclusion on the condition that the number of sources and the round trip delay time are both small. Under these circumstances, FERM performs better than a threshold based non-fuzzy control strategy. However, under these circumstances most controllers provide satisfactory results, and what is required is a robust controller capable of graceful degradation in the presence of greater delays and more connections. The results presented here, indicate that FERM is less robust in these cases than the threshold based controller.

The FERM algorithm could be modified to improve its performance in the in the conditions simulated here. Two possible approaches would be to adjust the rules, and to adjust the fuzzification/defuzzification process. However, manually adjusting the algorithm to the particular operating condition may not lead to a robust algorithm suitable for general application. Several of these approaches are explored in [10]. Further research will be needed before it is possible to say whether or not fuzzy control will be a suitable solution to the problem of ABR rate control.

Fig. 9. Maximum buffer occupancy vs RTT for one to five ABR sources. Buffer capacity is 1024 cells.



APPENDIX

A linguistic values in FERM is defined by four values (x_1, x_2, x_3, x_4). These values define the way in which fuzzification and defuzzification are performed. 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 membership in the set moderate, e.g., the truth value of the statement "100 is moderate" is 0. An input value in the range (380, 550) has unity membership in the set, and so 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, and membership decreases between x_3 and x_4, so that "685 is moderate" also 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 II.

The linguistic rules defined in terms of these variable and values are given in Table III.

TABLE II

LINGUISTIC VARIABLES AND THEIR POSSIBLE LINGUISTIC VALUES.

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 III

LINGUISTIC RULES FOR FERM.

if (q is empty)	and (dq is down_fast)	then rate := v_high
if (q is empty)	and (dq is down_slow)	then rate :=v_high
if (q is empty)	and (dq is zero)	then rate :=v_high
if (q is empty)	and (dq is up_slow)	then rate :=v_high
if (q is empty)	and (dq is up_fast)	then rate :=v_high
if (q is moderate)	and (dq is down_fast)	then rate :=v_high
if (q is moderate)	and (dq is down_slow)	then rate :=high
if (q is moderate)	and (dq is zero)	then rate :=moderate
if (q is moderate)	and (dq is up_slow)	then rate :=little
if (q is moderate)	and (dq is up_fast)	then rate :=v_little
if (q is full)	and (dq is down_fast)	then rate :=moderate
if (q is full)	and (dq is down_slow)	then rate :=little
if (q is full)	and (dq is zero)	then rate :=v_little
if (q is full)	and (dq is up_slow)	then rate :=v_little
if (q is full)	and (dq is up_fast)	then rate :=v_little

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.

[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 modelling 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] G.Ramamurthy and Q. Ren, "Analysis of the adaptive rate control for ABR service in ATM networks", *Proc. Globecom 95*, pp. 1083–1088, 1995.

[9] N. Yin "Analysis of a rate-based traffic management mechanism for ABR service", *Proc. Globecom 95*, pp. 1076–1082, 1995.

[10] L. Andrew and Kuan S.-H., "", to appear.