1998 International Computer Symposium
Workshop on Artificial Intelligence
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

A NEURAL-FUZZY CONGESTION CONTROLLER FOR ATM NETWORKS

Chun-Liang Hou and Shie-Jue Lee

Department of Electrical Engineering
National Sun Yat-Sen University, Kaohsiung, Taiwan, R.O.C.
Email:{hjl,leesj}@water.ee.nsysu.edu.tw

ABSTRACT

We propose the use of a neural-fuzzy scheme for a rate-based feedback controller in ATM (Asynchronous Transfer Mode) networks. ABR (Available Bit Rate) traffic is not guaranteed quality of service (QoS) in the setup connection, and it can dynamically share the available bandwidth. Therefore, congestion can be controlled by regulating the source rate to a certain degree depending on the current traffic flow. Traditional methods perform congestion control by monitoring the queue length. The source rate is decreased by a fixed rate when the queue length is greater than a prespecified threshold. However, it is difficult to get a suitable rate according to the degree of traffic congestion. We employ a neural-fuzzy mechanism to control the source rate. Through learning, membership values can be generated and cell loss can be predicted from the current queue length. Then an explicit rate is calculated and the source rate is controlled appropriately. Simulation results have shown that our method provides a better adaptive capability and a higher throughput than traditional methods.

Keywords: traffic control, cell loss, ATM, neural-fuzzy networks, fuzzy logic, cell rate

1. INTRODUCTION

Asynchronous transfer mode (ATM) is a modern technology enabling the integration of different traffic types within a single communication network. ATM networks are being widely developed to carry voice, video and data traffics. Various service classes have been defined in ATM for the support of traffic with different quality-of-service (QoS) requirements. These classes consist of constant bit rate (CBR), real-time variable rate (rt-VBR), non-real time variable bit rate (nrt-VBR), available bit rate (ABR), and unspecified bit rate (UBR). Available bit-rate (ABR) allows applications to fully utilize the available bandwidth in the network by adjusting their instantaneous transmission rates to the available capacity. The quality (QoS) of transmission cannot be determined in the connection setup time for ABR. Therefore, there is a need of a congestion management scheme to adaptively control the rate of transmission by closed-loop. A congestion control scheme is essential for the support of ABR traffic to utilize the available network bandwidth without causing congestion. Among various control schemes, the rate-based congestion control framework is widely used.

A rate-based flow-control scheme is an end-to-end feedback mechanism. The scheme consists of one source end system and one destination end system for each feedback loop. The source starts to send data cells with the rate of ICR (initial cell rate), and subsequent traffic flow is regulated by resource management (RM) cells. Each RM cell is allowed to flow all the way to the destination end system, and travels in the reverse direction. Intermediate switches mark down the rate in the reverse RM cells. The smallest allocation is therefore the value in the cell when it reaches the source. The source may then use this rate for subsequent transmission until a new resource management cell is received.

Many rate-based control schemes [1, 2, 3, 4, 5, 6, 7, 8] have been proposed. In the EFCI marking scheme, when a data cell from the source reaches a switch which is under congestion, the EFCI field of the cell header is set to one. If the destination checks that the EFCI field is one, it will send an RM cell back to the source. When the source receives an RM cell, it reduces the cell rate; otherwise the cell rate is increased. However, difficulties may arise if RM cells are lost or delayed. To solve this disadvantage, Branhart proposes the PRCA (proportional rate control algorithm) scheme. The source sends a data cell, in a constant rate, in which EFCI is set to zero. When the destination checks that EFCI is zero, it sends an RM cell back to the source and the source is informed that the cell rate can be increased. The source will decrease the cell rate when RM cells are not received in the expected period. Therefore, the source continues to decrease the cell rate if RM cells are lost or delayed. However, a long path VC has a large probability of EFCI bit being set; a switch under congestion sets EFCI on all VCs. Roberts [9] proposed EPRCA (enhanced proportional rate control algorithm). Congestion is detected at switches based on the queue length. When the queue length is greater than a prespecified threshold, the source is asked to reduce the cell rate by a fixed rate. However, it is difficult to get a suitable rate according to the degree of traffic congestion. Furthermore, this scheme may cause a high cell loss rate.

We propose a neural-fuzzy rate-based feedback controller. We employee a neural-fuzzy mechanism to control the source rate. By monitoring the current queue state and the rate of change of the queue length, the future queue behavior is predicted. This closed-loop control scheme can maximize the traffic flow and avoid congestion. Simulation results have shown that our method provides a better adaptive capability and a higher throughput than EPRCA.

The rest of the paper is organized as follows. Section 2 provides a brief introduction to neural-fuzzy networks. An overview of the system is given in Section 3. Section 4 describes the prediction of cell loss by a neural-fuzzy network. Section 5 presents a self-tuning fuzzy inference engine. Simulation results are given in Section 6. Finally, conclusions are summarized in Section 7.

2. NEURAL-FUZZY NETWORKS

Neural-fuzzy networks borrow the ideas from fuzzy logic and neural networks. They have many advantages, such as learning abilities, optimization abilities, human-like IF-THEN rule thinking, and ease of incorporating expert knowledge. It is being used widely for various applications.

Neural-fuzzy networks can be divided into two kinds. One can construct the structure from a set of input-output training data pairs through learning. The other consists of a predefined architecture determined previously by some other means. Of course, the first kind is more desirable. However, it is not easy to get proper fuzzy partitioning from the input-output data.

Lin & Lin [10] proposed a fuzzy adaptive learning control network (FALCON) which can automatically construct fuzzy partitioning by learning from training examples. The FALCON is a five-layer structure. Layer one is the input layer, containing input nodes (input linguistic variables). The output of layer one is linked directly to layer two which is called the term layer and acts as membership functions representing the terms of the respective linguistic variables. Layer three contains rule nodes which represent fuzzy rules. Layer-three links present the preconditions of the rule nodes. Nodes at layer four are called output-term nodes, representing membership functions. Each layer-4 node represents a term of an output-linguistic variable. Layer five is the output layer. Each node in this layer is called an output-linguistic node and corresponds to one output-linguistic variable. FALCON does not allow the slope of trapezoidal functions to be adjustable. This may affect the speed of learning. Also, the max layer of layer-4 is not suitable to rule extraction.

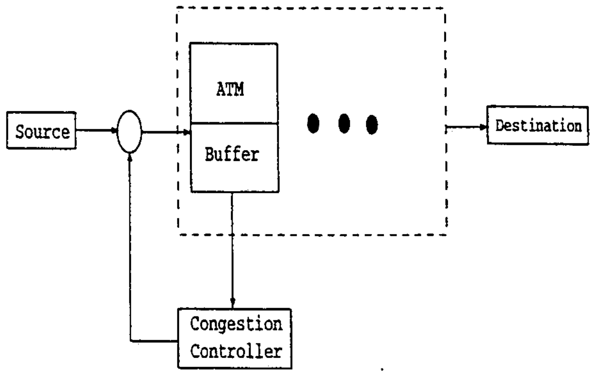

Figure 1: Congestion control in ATM networks.

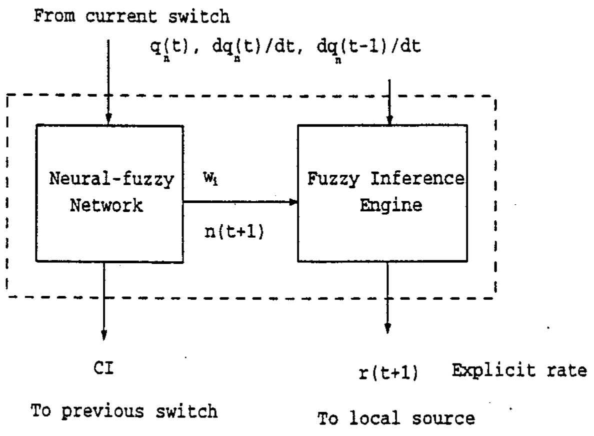

Figure 2: The block diagram of our neural-fuzzy controller.

3. SYSTEM OVERVIEW

The goal of congestion control is to smooth down the burstiness of the input traffic to avoid congestion. As shown in Figure 1, data cells are sent from the source to the destination through a series of switches. For the sake of simplicity, we only show one switch in Figure 1. Congestion controllers monitor the status of switches and notify the source to increase or decrease the rate of transmission, in order to maximize the throughput and avoid congestion.

Our congestion controller consists of two parts, a neural-fuzzy network and a fuzzy inference engine, as shown in Figure 2. The neural-fuzzy network provides the prediction of the future cell loss, while the fuzzy inference engine calculates an explicit rate to regulate the cell rate of the source. The neural-fuzzy network predicts cell loss by monitoring the current normalized queue length $q_n(t)$, normalized queue growth $dq_n(t)/dt$, and the previous normalized queue growth $dq_n(t-1)/dt$. Fuzzy rules and membership values are obtained by learning. These rules and membership functions are passed to the fuzzy inference engine, shown by the symbol w_i. Another parameter, cell loss $n(t+1)$,

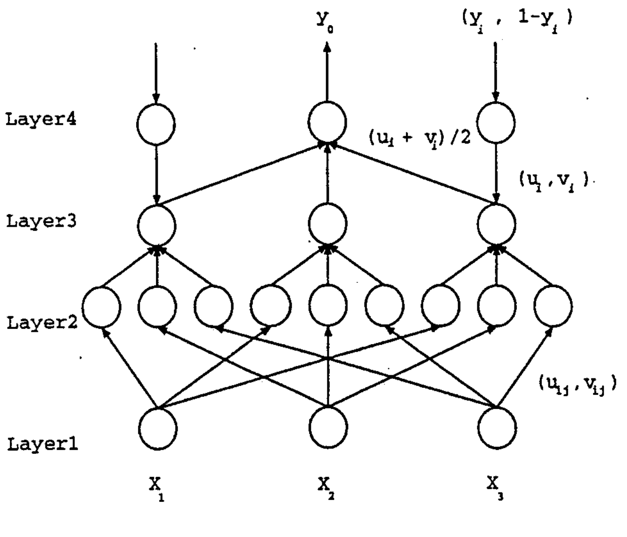

Figure 3: The architecture of our neural-fuzzy network.

is also predicted and passed to the fuzzy inference engine. A performance function is used to maintain the queue length. If the future cell loss is greater than one, the switch sets *CI* (congestion indicator) and notifies the previous switch in an RM cell that passes through the switch. Also, the switch calculates an explicit rate $r(t+1)$ to adjust the local source. Therefore, the system is a closed-loop scheme between sources and switches.

4. CELL LOSS PREDICTION

As mentioned, we use a neural-fuzzy network to predict cell loss based on the current normalized queue length $q_n(t)$, the current normalized queue change rate $dq_n(t)/dt$, and the previous normalized queue change rate $dq_n(t-1)/dt$. Learning of the neural-fuzzy network can be divided into two phases. In the first phase, rule nodes are constructed. Fuzzy ART is used to obtain the structure of the network. In the second phase, parameters are adjusted by a back-propagation algorithm.

The architecture of the neural-fuzzy network contains four layers, as shown in Figure 3, in contrast to five layers of FALCON described in Section 2. Note that there are three input nodes in Layer 1. The inputs to Layer 1 are x_1, x_2, and x_3, representing $q_n(t)$, $dq_n(t)/dt$, and $dq_n(t-1)/dt$, respectively. The function of each layer is described below.

1. Layer 1. Nodes in this layer just transmit input signals and their complements to the next layer directly. The output function uses complement coding of fuzzy ART:

$$f_i^{(1)}(x_i, x_i^c) = (x_i, 1 - x_i). \tag{1}$$

Note that each input node has two output values, x_i and 1-x_i.

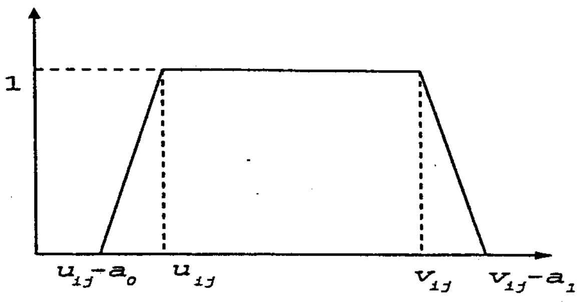

Figure 4: The membership function.

2. Layer 2. Each node of this layer represents a term of an input-linguistic variable. The membership function is a trapezoidal function, as shown in Figure 4, and can be expressed as follows

$$f_i^{(2)} = \begin{cases} \frac{x_i - u_{ij}}{a_0} + 1 & u_{ij} - a_0 < x_i \le u_{ij} \\ 1 & u_{ij} < x_i \le v_{ij} \\ \frac{v_{ij} - x_i}{a_1} + 1 & v_{ij} < x_i \le v_{ij} + a_1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Note that in contrast to FALCON, we allow the slopes of the two sides, i.e., a_0 and a_1, of a trapezoidal function to be trainable.

3. Layer 3. This layer is called the rule layer and each node in this layer represents one fuzzy logic rule. The output of a rule node is calculated by the product operation as follows

$$f_i^{(3)} = \prod_{i=1}^{n} f_i^{(2)}. \tag{3}$$

4. Layer 4. This layer is called output-linguistic layer and contains two kinds of nodes, training nodes and output nodes. Training nodes feed training data downward to the network. Their activation function is similar to that of layer 1, namely,

$$f_i^{(4)} = (y_i, 1 - y_i) \tag{4}$$

where y_i is the desired output. Output nodes present output values to the outside world. They act as defuzzifier. The activation function is defined to be the center of area:

$$y_{out} = f_i^{(4)} = \frac{\sum_i \frac{(u_i + v_i)}{2} f_i^{(3)}}{\sum_i f_i^{(3)}} \tag{5}$$

where u_i and v_i are the end points of a range within which values are allowed to exist.

As described earlier, the first learning phase concerns the construction of the network structure. We employ the ideas of fuzzy ART to do this job. Training patterns are fed into the input layer. Let X be the output vector of Layer 1 corresponding to a training pattern. That is, X contains the training pattern and its complement. We compute the following value

$$T_j(X) = \frac{|X \bigwedge W_j|}{\alpha + |W_j|} \tag{6}$$

where W_j is defined as

$$W_j \equiv \{u_{ij}, 1 - v_{ij}\}. \tag{7}$$

The value denotes the similarity between the input pattern and W_j. Let

$$T_J = max\{T_1, T_2, \ldots, T_N\} \tag{8}$$

Then, T_J is the winner and we do vigilance test

$$\frac{|X \bigwedge W_J|}{|X|} \geq \rho \tag{9}$$

for category node J. If vigilance test fails, then mismatch reset occurs. The category will be disabled and the process restarts to search for the next category until a match is obtained. If no such node is found, a new input-linguistic node and a corresponding training node are created. When a category node passes vigilance test, we have to perform vigilance test for the desired output. If the desired output fails, a new input-linguistic node and a corresponding training node are created at the same time. Otherwise, we modify the weights by taking the AND operation between weights and input patterns.

When the structure of the network is set, we proceed to do the second phase of learning. In this phase, we intend to minimize errors with the gradient descent method, by adjusting the parameters associated with membership functions. The error function is defined to be

$$E = \frac{1}{2}(y_d - y_o)^2, \tag{10}$$

where y_d is the desired output value and y_o is the actual output value. Obviously, only Layer 2 and Layer 4 have parameters associated with them. These parameters control the shape of a trapezoidal membership function. Therefore, no tuning is needed for Layer 1. However, we have to consider Layer 3 since we use back-propagation to transmit error signals from Layer 4 to Layer 2. We describe the learning process for layers 4, 3, and 2 as follows.

- Layer 4. In this layer, values are allowed to exist between u_i and v_i. Therefore, we only adjust these two parameters. Take the following gradient:

$$\frac{\partial E}{\partial u_i} = \frac{\partial E}{\partial f_i^{(4)}} \frac{\partial f_i^{(4)}}{\partial u_i} = -(y_d - y_o)\frac{f_i^{(3)}}{2\sum_i f_i^{(2)}}. \tag{11}$$

Hence, u_i is updated by

$$u_i(t+1) = u_i(t) + \eta(y_d - y_o)\frac{f_i^{(3)}}{2\sum f_i^{(3)}} \tag{12}$$

where η is the learning rate. Similarly, v_k is updated by

$$v_i(t+1) = v_i(t) + \eta(y_d - y_o)\frac{f_i^{(3)}}{2\sum_i f_i^{(3)}}. \tag{13}$$

- Layer 3. The error signal can be computed by

$$\delta = \frac{-E}{\partial f_i^{(4)}} \frac{\partial f_i^{(4)}}{\partial f_i^{(3)}} \tag{14}$$

and can be expressed as

$$\begin{aligned}\delta &= (y_d - y_o) \times \\ &[\frac{\frac{(u_k+v_k)}{2} \times \sum_k f_k^{(3)} - \sum_k \frac{(u_k+v_k)}{2} \times f_k^{(3)}}{[\sum_k f_k^{(3)}]^2}].\end{aligned} \tag{15}$$

This signal will be used in Layer 2.

- Layer 2. In this layer, four parameters u_{ij}, v_{ij}, a_1, and a_0 of each membership function have to be trained. Take the following gradient:

$$\frac{\partial E}{\partial u_{ij}} = -\frac{\partial E}{\partial f_j^{(3)}} \frac{\partial f_j^{(3)}}{\partial f_i^{(2)}} \frac{\partial f_i^{(2)}}{\partial u_{ij}} \tag{16}$$

where

$$\frac{\partial f_i^{(2)}}{\partial u_{ij}} = \begin{cases} \frac{-1}{a_0} & u_{ij} - a_0 < x_i \leq u_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Hence, u_{ij} is updated by

$$u_{ij}(t+1) = u_{ij}(t) + \eta\frac{\partial f_i^{(2)}}{\partial u_{ij}}\delta \prod_{k \neq i} f_k^{(3)}. \tag{18}$$

Similarly, we have

$$v_{ij}(t+1) = v_{ij}(t) + \eta\frac{\partial f_i^{(2)}}{\partial v_{ij}}\delta \prod_{k \neq i} f_k^{(3)} \tag{19}$$

$$a_0(t+1) = a_0(t) + \eta\frac{\partial f_i^{(2)}}{\partial a_0}\delta \prod_{k \neq i} f_k^{(3)} \tag{20}$$

$$a_1(t+1) = a_1(t) + \eta\frac{\partial f_i^{(2)}}{\partial a_1}\delta \prod_{k \neq i} f_k^{(3)} \tag{21}$$

and

$$\frac{\partial f_i^{(2)}}{\partial v_{ij}} = \begin{cases} \frac{1}{a_1} & v_{ij} < x_i \leq v_{ij} + a_1 \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

$$\frac{\partial f_i^{(2)}}{\partial a_0} = \begin{cases} \frac{-1}{a_0} \times (x_i - u_{ij}) & u_{ij} - a_0 < x_i \leq u_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

$$\frac{\partial f_i^{(2)}}{\partial a_1} = \begin{cases} \frac{1}{a_1} \times (x_i - v_{ij}) & v_{ij} < x_i \leq v_{ij} + a_1 \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

5. FUZZY INFERENCE ENGINE

The purpose of control is to influence the behavior of a system by changing inputs to that system according to a set of rules that model how the system operates. Classic control theory uses a mathematical model to define a relationship that transforms the desired state (requested) and observed state (measured) of the system into inputs that will alter the future state of that system. The major drawback of this approach is that it usually assumes that the system being modeled is linear or at least behaves in some fashion like a monotonic function. As the complexity of the system increases it becomes more difficult to formulate that mathematical model. Fuzzy control replaces the role of the mathematical model and replaces it with another that is built from a number of smaller rules that in general only describe a small section of the whole system. The process of inference binding them together to produce the desired outputs. That is, a fuzzy model has replaced the mathematical one. The inputs and outputs of the system have remained unchanged.

We use a self-tuning fuzzy inference engine for the congestion controller, to control the traffic flow and make the network's available bandwidth maximally utilized. This model was proposed in [11]. Our method is to minimize the difference between the current buffer length and the desired buffer length. Therefore, the desired buffer length can be maintained and buffer overflow can be avoided.

Fuzzy inference is based on a number of fuzzy IF-THEN rules. These rules are created automatically from our neural-fuzzy network by learning. Each rule R_i is expressed as

IF $q_n(t)$ is V_{1i} and $dq_n(t)/dt$ is V_{2i} and $dq_n(t-1)/dt$ is V_{3i} THEN $Y_i(r(t), I(t))$

where, as before, $q_n(t)$ is the current normalized queue length, $dq_n(t)/dt$ is the current normalized queue change, and $dq_n(t-1)/dt$ is the previous queue change. $I(t)$ is the vector consisting of $q_n(t)$, $dq_n(t)/dt$, and $dq_n(t-1)/dt$. Note that $Y_i(r(t), I(t))$ is the action function and is defined as

$$\begin{aligned} Y_i(r(t).I(t)) &= r(t) + A_i(q_d - q(t)) \\ &\quad -B_i(dq_n(t)/dt - 0.5) \\ &\quad -C_i(dq_n(t-1)/dt - 0.5) \\ &\quad -D_i(n(t+1)). \end{aligned} \tag{25}$$

Finally, the system can calculate the rate of cell transmission by the following equation:

$$r(t+1) = \frac{\sum_i Y_i \times w_i}{\sum_i w_i} = \frac{\sum_i Y_i \times w_i}{w} \tag{26}$$

where $r(t+1)$ is the explicit rate that will be sent back to the source to adjust the transmutation rate, and w_i is the weight of the ith rule and is defined as

$$\begin{aligned} w_i &= V_{1i}(q_n(t)) \times V_{2i}(dq_n(t)/dt) \\ &\quad \times V_{3i}(dq_n(t-1)/dt) \end{aligned} \tag{27}$$

where $V_{mi}(X)$ denotes the membership degree of X on V_{mi}.

In order to avoid congestion, the system is designed to minimize a performance function based on the gradient descent method. The performance function is defined as:

$$P(t) = \frac{(q(t) - q_d)^2}{2} \tag{28}$$

where q_d is the desired queue length. The current queue length, $q(t)$, can be expressed as:

$$\begin{aligned} q(t+1) &= q(t) + (MCR + \alpha(t) \times r(t+1) - O(t)) \\ &\quad \times T \end{aligned} \tag{29}$$

where $O(t)$ is the output rate, T is the interval time, MCR is the other VC's mean cell rate and $\alpha(t)$ denotes the transmission demand, defined as:

$$\alpha(t) = \frac{CMR}{r(t)} \tag{30}$$

where CMR is the current measured rate and $r(t)$ is the previous feedback rate. The system calculates the difference between the current queue length and the desired queue length. From the calculated difference, the parameters A_i, B_i, C_i, and D_i used in Eq.(25) can be calculated. For example, A_i can be obtained as follows:

$$A_i(t+1) = A_i(t) + \frac{\partial P(t+1)}{\partial A_i(t)} \tag{31}$$

and

$$\frac{\partial P(t+1)}{\partial A_i(t)} = (q(t+1) - q_d) \times \frac{\partial q(t+1)}{\partial A_i(t)}, \tag{32}$$

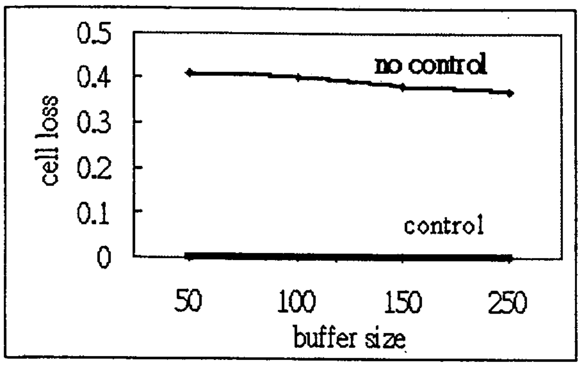

Figure 5: Cell loss rate vs. buffer size.

$$\frac{\partial q(t+1)}{\partial A_i(t)} = \frac{\partial q(t)}{\partial a_i(t-1)} + T \times \alpha(t) \times \frac{\partial r(t+1)}{\partial a_i(t)}, \quad (33)$$

$$\begin{aligned}\frac{\partial r(t+1)}{\partial a_i(t)} &= \sum_m [\frac{\partial r(t+1)}{\partial Y_m} \times \frac{\partial Y_m}{\partial a_i(t)}] \\ &= [\sum_m \frac{w_m}{w}] \times \frac{\partial r(t)}{\partial a_i(t-1)} \\ &\quad + \frac{w_i}{w} \times (q_d - q(t)). \qquad (34)\end{aligned}$$

Note that the initial conditions are defined as:

$$\frac{\partial r(1)}{\partial a_i(0)} = 0, \quad \frac{\partial q(1)}{\partial a_i(0)} = 0. \qquad (35)$$

B_i, C_i, and D_i can be calculated in the same way.

6. EXPERIMENTAL RESULTS

In this section, we show some experimental results. A comparison with EPRCA is also made.

First of all, we run an experiment to show the power of our controller in reducing the cell loss rate. In this experiment, one switch and two attached sources are simulated. The result is shown in Figure 5. We can see that without the controller, the cell loss rate is high. However, using the controller cell loss is nearly zero.

In the second experiment, we use a model shown in Figure 6 in which four ATM switches are cascaded [12]. Two sources, S1 and S2, are connected to Switch 1; One source, S3, and one destination, D1, are connected to Switch 2; One source, S4, is connected to Switch 3; Three destinations, D2, D3, and D4, are connected to Switch 4. Values for related parameters about sources are listed in Table 1. In this table, ICR stands for "initial cell rate", PCR for "peak cell rate", MCR for "mean cell rate", and AIR for "additive increase rate". Values for related parameters about switches are listed in Table 2. In this table, QT stands for "queue threshold", DQT for "queue very congested threshold", MRF for "major reduction factor", and ERF for "explicit reduction factor". We use the interrupted Bernoulli Process (IBP) as the traffic type for sources. The IBP state diagram is shown in Figure 7 in which P1, P2, P3, and P4 are transfer probabilities. The states, 0 and 1, represent the idle state and the burst state, respectively. Figure 8 shows the allowed cell rate (ACR) for EPRCA. This figure shows that ACR oscillates, meaning that EPRCA fails to have an adaptive capability in traffic control. Figure 9 shows the allowed cell rate (ACR) obtained from our method. This figure shows a very much smoother curve, meaning that our method has a strong adaptive capability in traffic control. Figure 10 compares the throughput of each switch. Obviously, our method results in a higher throughput at each switch.

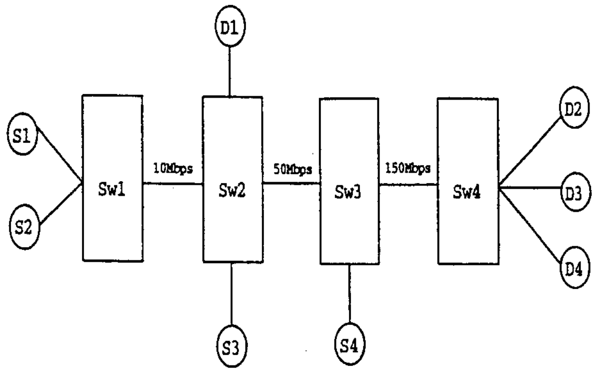

Figure 6: A network model.

Table 1: Parameter values for sources.

	S1	S2	S3	S4
ICR	3Mbps	3Mbps	5Mbps	5Mbps
PCR	150Mbps			
MCR	3Mbps			
AIR	10Mbps			

In the third experiment, we use different P_1 values to compare the throughput of our method and that of EPRCA for each switch. The simulation result is shown in Figure 11 which demonstrates that our method has a higher throughput, especially when the system is busy, i.e., P1 is high.

7. CONCLUSION

An ATM congestion controller based on a neural-fuzzy network and a fuzzy inference engine has been presented. The neural-fuzzy network predicts cell loss and

Table 2: Parameter values for switches.

	Sw1	Sw2	Sw3	Sw4
(QT, DQT)	40,45	80,90	80,90	80,90
MRF	0.725	0.825	0.825	0.825
ERF	0.325	0.625	0.625	0.625
Buffer size	50	100	100	100

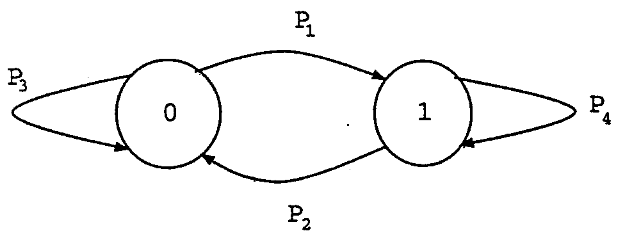

Figure 7: The IBP state diagram

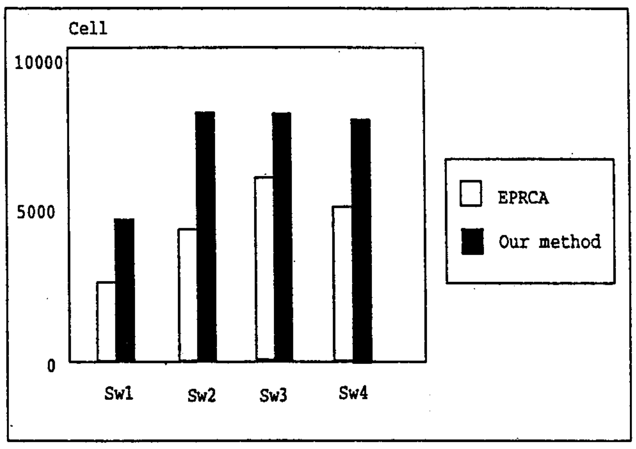

Figure 10: Throughput of each switch.

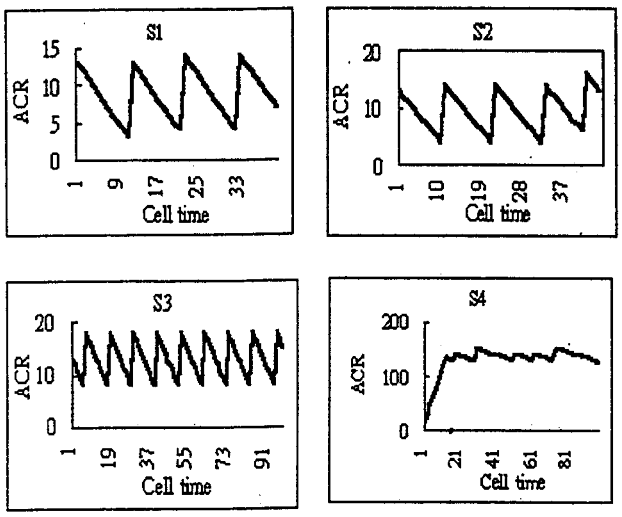

Figure 8: Allowed cell rate with EPRCA.

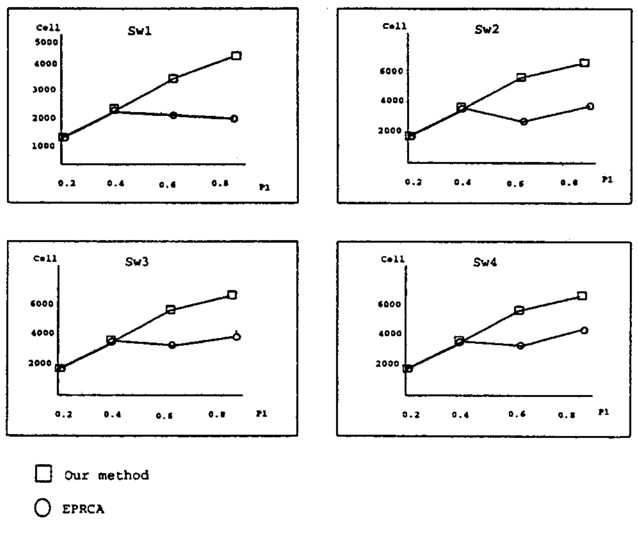

Figure 11: Throughput vs. different P1 values.

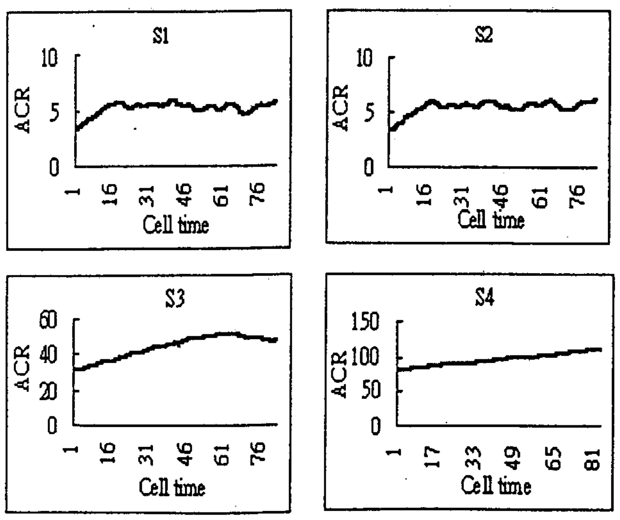

Figure 9: Allowed cell rate with our method.

derives fuzzy rules for the fuzzy inference engine. The fuzzy inference engine then calculates an explicit rate which can be used to control the cell rate of the sources. This closed-loop control scheme can maximize the traffic flow and avoid congestion. Simulation results have shown that our method increases throughput and reduces cell loss rate.

8. REFERENCES

[1] A. Atai, "A rate-based feedback traffic control in B-ISDN," *Proceedings of IEEE International Conference on Communications (ICC'94)*, pp. 1605–1615, 1994.

[2] R. G. Cheng and C. J. Chang, "Design of a fuzzy traffic controller for ATM networks," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 460–469, Nov 1996.

[3] L. Benmohamed and S. M. Meerkov, "Feedback control of congestion in packet switching networks," *IEEE/ACM Transactions on Networking*, vol. 6, Dec 1993.

[4] Y. C. Liu, "Rate regulation with feedback controller in ATM network - A neural network approach," *IEEE Journal on Selected Area in Communciations*, vol. 15, pp. 200–208, Feb 1997.

[5] A. A. Tar and I. W. Habib, "Congestion control mechanism for ATM networks using neural network," *Proceedings of IEEE International Conference on Communications (ICC'95)*, pp. 206–210, 1995.

[6] R. Durand and Y. C. Liu, "Neural network based congestion control for single and multiple sources in ATM networks," *Proceedings of IEEE/MACS Computer Engineering for System Applications*, pp. 1100–1105, July 1996.

[7] Y. C. Liu and C. Douligeris, "Rate regulation with feedback controller in ATM networks," *IEEE Journal on Selected Area in Communciations*, 1997.

[8] *The ATM Forum Technical Committee, "Traffic management specificatin version 4.0"*, 1996.

[9] L. Roberts, "Enchanced PRCA (proportional rate-control algorithm)," *Technical Report AF-TM*, pp. 154–157, 1994.

[10] C. J. Lin and C. T. Lin, "An ART-based fuzzy adaptive learning control network," *IEEE Transactions on Fuzzy Systems*, vol. 5, pp. 477–496, Nov 1997.

[11] Q. Hu and D. W. Petr, "Self-tuning fuzzy traffic rate control network," *Proceedings of IEEE International Conference on Communications (ICC'96)*, Dallas, TX, pp. 424–428, 1996.

[12] W. G. Lai, "A flow control scheme on ATM networks with max-min fairness," *Technical Report*, 1997.