Intelligent Leaky Bucket Algorithms for Sustainable-Cell-Rate Usage Parameter Control in ATM Networks

Chung-Ju Chang, *Senior Member, IEEE*, Chung-Hsun Yu, Chih-Sheng Chang, and Li-Fong Lin, *Student Member, IEEE*

***Abstract*—In this paper, we propose two intelligent leaky bucket algorithms for sustainable-cell-rate usage parameter control of multimedia transmission in asynchronous transfer mode networks. One is the fuzzy leaky bucket algorithm, in which a fuzzy increment controller (FIC) is incorporated with the conventional leaky bucket algorithm; the other is the neural fuzzy leaky bucket algorithm, where a neural fuzzy increment controller (NFIC) is added with the conventional leaky bucket algorithm. Both the FIC and the NFIC properly choose the long-term mean cell rate and the short-term mean cell rate as input variables to intelligently determine the increment value. Simulation results show that both intelligent leaky bucket algorithms have significantly outperformed the conventional leaky bucket algorithm, by responding about 160% faster when taking control actions against a nonconforming connection while reducing as much as 50% of the queueing delay experienced by a conforming connection. In addition, the neural fuzzy leaky bucket algorithm outperforms the fuzzy leaky bucket algorithm, in aspects of three performance measures such as selectivity, responsiveness, and queueing delay, especially when the traffic flow is bursty, dynamic, and nonstationary.**

***Index Terms*—Asynchronous transfer mode (ATM), fuzzy logic, leaky bucket algorithm, neural fuzzy, sustainable cell rate, usage parameter control.**

I. Introduction

THE EMERGENCE of multimedia services has diversified the quality-of-service (QoS) and bandwidth requirements for communication services. Asynchronous transfer mode (ATM) is considered as a suitable technique to meet the diverse requirements. Several traffic-control mechanisms are recommended for ATM networks [1]. Among them, call admission control (CAC) and usage parameter control (UPC) are the most important.

CAC is performed at the call setup phase of a new call to decide whether the call can be accepted or not. It accepts the call if the required bandwidth and QoS of the call can be afforded while QoS of existing connections can still be maintained. A traffic contract, which specifies traffic descriptors such as the *peak cell rate* (PCR), the *sustainable cell rate* (SCR), and the *maximum burst size* (MBS) would be made between the accepted connection and the network. For CAC to perform correctly, all the established connections must not violate their respective traffic contracts which are of vital importance to the decision making of CAC. To make sure that the established connections conform to their traffic contracts, CAC is coupled with UPC.

UPC is performed at the user-network interface during the data transfer phase. It is defined as the set of actions taken by the network to police the offered traffic of a connection so that the negotiated traffic contract is respected. Its main purpose is to protect network resources from malicious as well as unintentional misbehavior which can affect the QoS of other already established connections. However, the wide variety of multimedia services with different traffic characteristics and QoS requirements makes UPC a difficult job. The difficulty lies in finding a simple, universal, and effective UPC scheme which is able to police any types of traffic ranging from video to data traffic. Several UPC schemes such as the jumping window, triggered jumping window, moving window, exponentially weighted moving average, and leaky bucket mechanisms were studied and compared [2]–[5]. The most popular and well-known policing scheme is the leaky bucket algorithm because of its simplicity and effectiveness.

Monitoring and controlling PCR of a connection is not difficult because we only have to determine if the peak emission interval is smaller than the reciprocal of the negotiated PCR, Λ_{PCR}. However, policing the SCR of a connection is much more complicated because the connection is eligible to transfer cells with a short-term mean rate higher than the negotiated SCR, Λ_{SCR}, as long as the long-term mean rate of the connection conforms to Λ_{SCR}. Therefore, we here concentrate on the SCR UPC.

In this paper, we assume that a traffic shaper (TS) is equipped within the customer premise equipment to regulate the cell stream of the traffic source so as to conform the negotiated SCR. The regulation is to alter the traffic characteristics of the cell stream to achieve a desired traffic shape. However, the consequence of the regulation would cause an increase in the mean cell transfer delay. The conjunction of TS and UPC, referred to as TS-UPC, should employ an identical scheme with same parameters settings so that any possible illegal cell that might have been detected as nonconforming by UPC will be detected ahead of time and saved in the queue by TS. In this way, the TS-UPC can guarantee zero cell-loss ratio at UPC for

Manuscript received October 11, 2001; revised January 7, 2003. This work was supported by the National Science Council, Taiwan, R.O.C., under Contracts NSC 88-2213-E-009-088 and NSC 89-2213-E-009-105, and by the Lee and MTI Center, National Chiao-Tung University, Taiwan. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Tak-shing Peter Yum.

The authors are with the Department of Communication Engineering, National Chiao-Tung University, Hsinchu 300 Taiwan, R.O.C. (e-mail: cjchang@cc.nctu.edu.tw; chyu@zyxel.com.tw; dole@cht.com.tw; kawai.cm85g@nctu.edu.tw).

Digital Object Identifier 10.1109/TMM.2004.834860

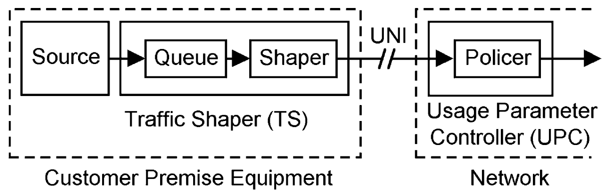


Fig. 1. Connection model.

a nonviolating connection. Nevertheless, if a user intentionally or unintentionally changes the parameters settings in TS and illegally enjoys a higher bit-rate service there, UPC will detect the violation and take actions against it.

Three performance requirements have to be fulfilled by TS-UPC in ATM networks.

1) *High selectivity:* UPC should detect and tag (drop) the nonconforming cells of a violating connection, as many as possible, while being transparent when the connection conforms to its traffic contract.
2) *High responsiveness:* the time for UPC to detect a violating connection should be rather short.
3) *Low queueing delay*: cells of a nonviolating connection should not experience too much queueing delay at TS. However, the queueing delay introduced by TS on a violating connection is beyond our consideration.

The primitive connection model with TS-UPC is shown in Fig. 1. The component attached to the traffic source is TS which contains a shaper and a queue. The shaper employs the leaky bucket algorithm to determine the conformance of cells. It bypasses the conforming cells but stores the nonconforming cells in the queue for further legal transmission. The component at the entrance of the network is UPC, where a policer is incorporated. The policer also employs the same leaky bucket algorithm as the shaper does. It bypasses the conforming cells but drops or tags the nonconforming cells.

The conventional TS-UPC using the leaky bucket algorithm recommended in ITU-T I.371 has a crisp structure with two fixed parameters of *threshold* and *increment*. It uses a parametric model for analysis, thus resulting in the lack of real multimedia traffic information which are dynamic, imprecise, nonlinear, and even nonstationary. Generally speaking, it is difficult for networks to acquire complete statistics of input traffic. Therefore, it is not easy to accurately determine the threshold or the increment in the multimedia traffic flows. The rationale and principles underlying the nature and choice of the threshold or the increment under dynamic and bursty conditions are unclear. As a result, the decision process of the network is based on incomplete information and full of uncertainty.

In this paper we propose two *intelligent* TS-UPCs using *fuzzy* and *neural fuzzy* leaky bucket algorithms to perform the sustainable-cell-rate UPC of multimedia transmission in ATM networks. The *fuzzy logic system* and *neural network* are both numerical model-free estimators and dynamical systems [6]. The fuzzy set theory appears to provide a robust mathematical framework for dealing with real-world imprecision. Its approach exhibits a soft behavior which has the capability to adapt to dynamic, imprecise, and bursty environments. The fuzzy logic system can represent information in a way that resembles natural human communication, and can handle the information in a way similar to human reasoning [7]–[9]. It would be an intelligent implementation that not only refers to the mathematical formulation of classical control but also mimics expert knowledge in traffic control. Neural networks are trainable systems that demonstrate the ability to learn, recall, and generalize from training patterns or data. Through learning, neural networks can predict nonlinear complex functions, thus making themselves effective tools to be employed in ATM networks for traffic modeling and prediction [6].

In recent years, neural network research has pursued either by a pre-structuring of the neural network to improve its performance, or by a possible interpretation of the synaptic matrix following the learning stage; and fuzzy logic research has pursued the development of methods for automatic tuning of the parameters which characterize the fuzzy control system. Notice that fuzzy set theory possesses no clear, general technique to map expert knowledge of traffic control onto the design parameters of the fuzzy logic controller (FLC). Hence, one approach that receives the benefits of neural networks and fuzzy logics and solves their respective problems, called the *neural fuzzy network*, is developed. The neural fuzzy network integrates the fuzzy logics within a neural network. The integration brings the low-level learning and computational power of the neural network into the fuzzy logic system, and provides the high-level, human-like thinking and reasoning of the fuzzy logic system for the neural network. The neural fuzzy network generally takes the form of a multilayer neural network to realize a fuzzy logic system. It is a *structured* neural network that can incorporate domain knowledge from conventional policies; and it not only provides a robust framework to mimic experts' knowledge embodied in existing traffic-control techniques, but also constructs intelligent computational algorithm for traffic control [6], [10].

Some literature has studied traffic policing using the intelligent techniques [11]–[14]. In [11], a neural network traffic-enforcement mechanism using window-based scheme for ATM networks was presented. It is based upon an accurate estimation of the probability density function (pdf) of traffic via count process, and the system performance is evaluated in terms of the pdf violation. It has scalability and convergence problems if the number of previous windows is needed to be large. In [12], a fuzzy controller for managing voice cells in ATM networks was proposed. Simulation results showed that the fuzzy leaky bucket had performance improvement over the conventional leaky bucket. In [13], the paper designed a fuzzy policer based on a window control scheme, which has the characteristics of simplicity and the capacity to combine a high degree of responsiveness with a selectivity close to that of an ideal policer. In [14], the proposed policing strategy integrated with a linear prediction filter is used to forecast the cell rate of the policed traffic source.

In this paper, one TS-UPC using a fuzzy logic system, called *fuzzy* TS-UPC, is designed to contain a fuzzy leaky bucket algorithm incorporated with a *fuzzy increment controller* (FIC) for dynamic increment adjustment. Two system parameters, the long-term mean cell rate and the short-term mean cell rate, of a connection are fed into the FIC to adaptively calculate the appropriate increment. Simulation results show that the fuzzy

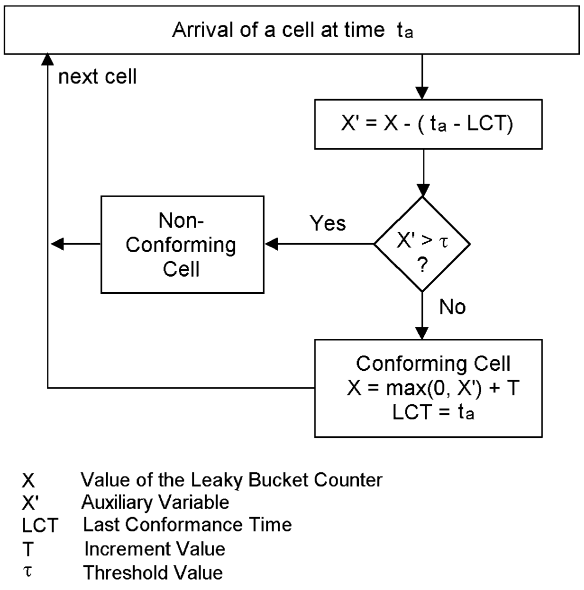


Fig. 2. Flowchart of the conventional leaky bucket algorithm.

TS-UPC can have higher selectivity, faster responsiveness, and smaller queueing delay than the conventional TS-UPC, as anticipated. The other TS-UPC using a neural fuzzy network, called the *neural fuzzy* TS-UPC, is designed to consist of a neural fuzzy leaky bucket algorithm incorporated with a *neural fuzzy increment controller* (NFIC) to dynamically adjust the increment. Neural fuzzy network is a *structured neural network*; it integrates intelligent learning and computation of neural networks with fuzzy logic systems. Also, the reinforcement learning is applied here for the NFIC, since we cannot measure the desired increment. Simulation results show that the neural fuzzy TS-UPC performs further better than the fuzzy TS-UPC in the above-mentioned performance measures of selectivity, responsiveness, and queueing delay, especially as the multimedia traffic flows are more bursty, dynamic, and nonstationary.

The paper is oriented as follows. In Section II, we provide an introduction of the leaky bucket algorithm recommended in ITU-T I.371 for conventional TS-UPC and the problems it encounters. In Section III, we describe the proposed fuzzy leaky bucket algorithm for the fuzzy TS-UPC. In Section IV, we describe the proposed neural fuzzy leaky bucket algorithm for the neural fuzzy TS-UPC. The performance measures of selectivity, responsiveness, and queueing delay for the conventional TS-UPC, the fuzzy TS-UPC, and the neural fuzzy TS-UPC are compared in Section V. Finally, some concluding remarks are presented in Section VI.

II. Leaky Bucket Algorithm

ITU-T Recommendation I.371 [1] recommends the Generic Cell Rate Algorithm (GCRA) as a conformance test for the cell stream of a connection. The GCRA has two equivalent versions—the virtual scheduling algorithm and the leaky bucket algorithm. The latter seems to be better comprehended since it can be pictured as a virtual leaky bucket whose content determines the conformance of a cell. As shown in Fig. 2, the leaky bucket is viewed as a finite capacity bucket whose real-valued content drains out at one unit rate but is increased by T units for each conforming cell. If a cell arrives at the time when the bucket content X' is above the threshold value τ, then the cell is nonconforming; otherwise, the cell is conforming and the bucket content is added by an increment T.

The conventional TS-UPC employs the leaky bucket algorithm in the shaper and the policer as their schemes to monitor the SCR of a connection. The threshold value τ is taken to be $\tau_{IBT}+\tau'_{SCR}$ and the increment T is taken to be the reciprocal of the negotiated SCR Λ_{SCR} of the connection, where τ_{IBT} is the intrinsic burst tolerance (IBT) used to limit the burst size to the negotiated MBS and τ'_{SCR} is an additional tolerance added to account for the cell delay variation (CDV) introduced by multiplexing schemes. Details of the two parameters τ_{IBT} and τ'_{SCR} can be found in the ITU-T Recommendation I.371.

If Λ_{SCR} is set to be the mean cell rate Λ_{mean} for the TS-UPC, then the possible rate fluctuations of the connection around the claimed mean cell rate will cause the leaky bucket within TS to detect some nonconforming cells. These detected nonconforming cells are stored in the queue, resulting in a long queueing delay. The undesirable long queueing delay can be avoided by making the bucket threshold τ in TS and UPC deviate from $\tau_{IBT} + \tau'_{SCR}$ to a large value. Unfortunately, a higher τ would cause the slower response time for UPC. Another solution without changing the bucket threshold is to make Λ_{SCR} be Λ_{mean} multiplied by a magnifying factor C, $C > 1$. By doing this, we can eliminate the retardation provoked by a higher τ. However, it has a risk of letting a connection with small rate fluctuations, e.g., a CBR connection, enjoy bandwidth higher than that negotiated. There are an infinite number of admissible couples of values for Λ_{SCR} and τ. The detailed analysis for the selection of Λ_{SCR} and τ and the consequent system performance can be found in [15].

III. Fuzzy Leaky Bucket Algorithm

The FLC has three functional blocks: a fuzzifier, a defuzzifier, and an inference engine with a fuzzy rule base [16], as shown in Fig. 3. The fuzzifier performs the function of fuzzification that translates the value of each input linguistic variable x into fuzzy linguistic terms. These fuzzy linguistic terms are defined in a term set and are specified by a set of membership functions. The defuzzifier describes an output linguistic variable of the control action y by a term set which is characterized by a set of membership functions, and adopts a defuzzification strategy to convert the linguistic terms into a nonfuzzy value that represents control action y. The term set should be determined at an appropriate level of granularity to describe the value of linguistic variables, and the number of terms in a term set is selected as a compromise between the computational complexity and the controlled performance. The fuzzy rule base is the control policy knowledge base, which is characterized by a set of linguistic statements in the form of "if-then" rules that describe the fuzzy logic

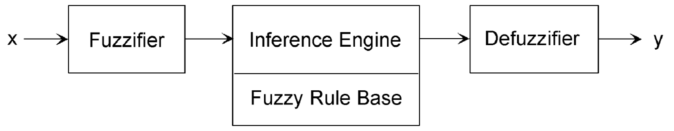
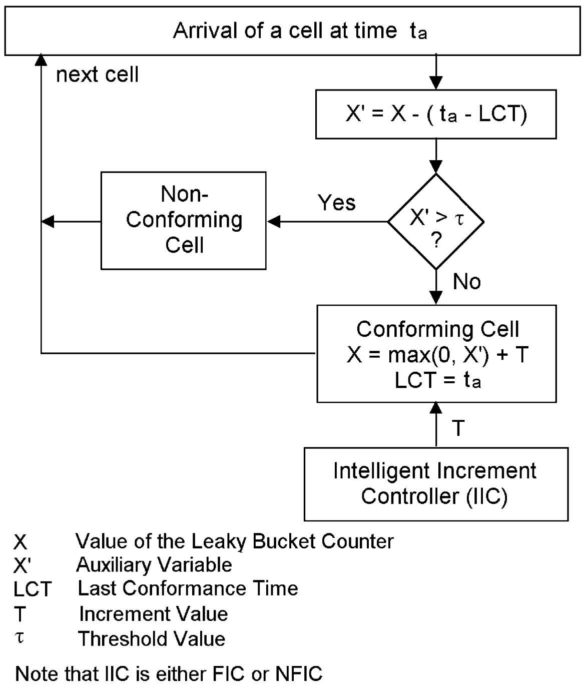


Fig. 3. Basic structure of a fuzzy logic controller.



Arrival of a cell at time ta
next cell
X' = X - (ta - LCT)
Non-Conforming Cell
Yes
X' > τ ?
No
Conforming Cell
X = max(0, X') + T
LCT = ta
T
Intelligent Increment Controller (IIC)
X Value of the Leaky Bucket Counter
X' Auxiliary Variable
LCT Last Conformance Time
T Increment Value
τ Threshold Value
Note that IIC is either FIC or NFIC

Fig. 4. Intelligent leaky bucket algorithm.

relationship between the input variables x and the control action y. The inference engine embodies the decision-making logic. It acquires the input linguistic terms of input linguistic variables from the fuzzifier and uses an inference method to obtain the output linguistic terms of output linguistic variables [7].

Fig. 4 shows an intelligent fuzzy leaky bucket algorithm which contains the conventional leaky bucket algorithm incorporated with an intelligent increment controller (IIC). The first intelligent leaky bucket algorithm we proposed is the fuzzy leaky bucket algorithm which employs a FIC to implement IIC. FIC is designed to dynamically adjust T, instead of using a fixed $T = 1/\Lambda_{SCR}$, so that the selectivity, responsiveness, and queueing delay can be optimally achieved. The reason we use the fuzzy logic system to implement the increment controller is because the fuzzy logic can represent information in a way resembling natural human communication and handle the information in a way similar to human reasoning [7]. The domain knowledge for the adjustment of T is as follows. When the cell stream of a connection appears to be violating the negotiated SCR, T should be adjusted to be big so that the leaky bucket can *quickly* detect the nonconforming cells; while in contrast, when the cell stream of a connection appears to be conforming or conservative to the SCR, T should be adjusted to be reasonably small so that no cell of the connection will be detected as nonconforming cells by the fuzzy leaky bucket (i.e., the leaky bucket would be transparent to the connection).

We choose two input variables for FIC: the long-term mean cell rate Λ_L and the short-term mean cell rate Λ_S of the connection being policed. The long-term mean cell rate is defined as the average cell rate of a connection since the beginning of the connection, and the short-term mean cell rate is defined as a moving-average cell rate in a time window. Λ_L and Λ_S are used to provide an indication of the conformance degree of the connection. At the arrival of a cell, the statistics of Λ_L and Λ_S are fed into FIC to obtain an optimal increment T.

We design Λ_L and Λ_S to have the same term set—$\{Low, Moderate, High\}$. And let T have five terms—$\{Very\ Small\ (T_1), Small\ (T_2), Medium\ (T_3), Big\ (T_4), Very\ Big\ (T_5)\}$. Fig. 5(a) and Fig. 5(b) shows the membership function of the input and output variables, respectively. The membership functions for $\{Low, Moderate, High\}$ are denoted by $\{A_{Lo}, A_{Mo}, A_{Hi}\}$; the membership functions for $\{T_1, T_2, T_3, T_4, T_5\}$ are denoted by $\{A_{T_1}, A_{T_2}, A_{T_3}, A_{T_4}, A_{T_5}\}$. These membership functions in the figures are represented by either triangular or trapezoidal functions, which have the advantage of simple computational complexity.

The rule base is designed according to the domain knowledge on how the FIC should behave. For example, the knowledge and experience tell us: when both Λ_L and Λ_S are lower than Λ_{SCR}, FIC should generate a very small T so that the connection can enjoy a higher cell rate later because the connection is likely to be too conservative; when both Λ_L and Λ_S are higher than Λ_{SCR}, the connection is likely to violate the negotiated SCR and FIC should generate a very big T so that the violation will be detected quickly. The inference rule base is shown in Table I. Below is an example of how the rules should be read.

Rule 1: If (Λ_L *is Low*) *and* (Λ_S *is Low*), *then* (T *is* $Very\ Small$).

The linguistic values of T_1, T_2, T_3, T_4, and T_5 of the output linguistic variable T are defined over a discrete universe of discourse having 65 536 points. The inference method adopts $\max - \min$ scheme. Take rule 1 and rule 2, which have the same term $Very\ Small\ (T_1)$ for example. In the first step, the $\max - \min$ inference method applies the $\min$ operator on membership values of associated term of all the input linguistic variables for each rule. We denote the firing strength of rule 1 and rule 2 by w_1 and w_2

$$w_1 = \min\left(A_{Lo}(\Lambda_L), A_{Lo}(\Lambda_S)\right) \tag{1}$$

$$w_2 = \min\left(A_{Lo}(\Lambda_L), A_{Mo}(\Lambda_S)\right). \tag{2}$$

Then applying the $\max$ operator between w_1 and w_2 yields the overall membership value of T_1, denoted by

$$w_{T_1} = \max(w_1, w_2). \tag{3}$$

The defuzzification method uses the *center of area* mechanism to obtain T

$$T = \frac{\sum_{i=1}^{n} A_T(z_i) * z_i}{\sum_{i=1}^{n} A_T(z_i)} \tag{4}$$

where n is the number of points of the output, $n = 65\,536$, z_i is the amount of control output at point i, and $A_T(z_i)$ represents its membership value in the output term set $\{T_1, T_2, T_3, T_4, T_5\}$ [7], which is given by

$$A_T(z_i) = \max_{j \in [1,5]} \left[\min\left(w_{T_j}, A_{T_j}(z_i)\right)\right]. \tag{5}$$

After FIC is built, the membership functions are finely tuned by observing the progress of simulation. The tuning can be done with different objectives, such as the response time and

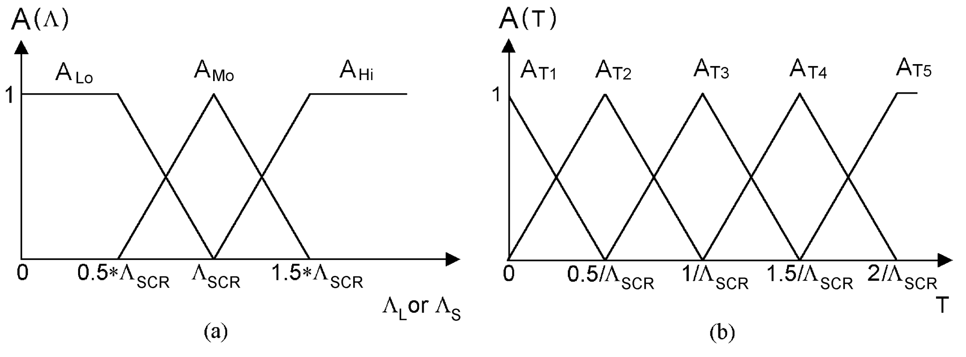


Fig. 5. (a) Membership functions for the input variables Λ_L and Λ_S. (b) Membership functions for the output variable T.

TABLE I
RULE BASE FOR FIC

Rule	Λ_L	Λ_S	T
1	Low	Low	Very Small (T_1)
2	Low	Moderate	Very Small (T_1)
3	Low	High	Small (T_2)
4	Moderate	Low	Small (T_2)
5	Moderate	Moderate	Medium (T_3)
6	Moderate	High	Big (T_4)
7	High	Low	Big (T_4)
8	High	Moderate	Very Big (T_5)
9	High	High	Very Big (T_5)

queueing delay. Any gain in response time must be traded off by a possible increase in the queueing delay experienced by a cell. However, since the tuning of the membership functions is intuitive, it is easy to achieve an appropriate balance between an acceptable queueing delay and a satisfactory responsiveness. The final control surface of FIC is shown in Fig. 6.

IV. NEURAL FUZZY LEAKY BUCKET ALGORITHM

The second intelligent leaky bucket algorithm we proposed is the neural fuzzy leaky bucket algorithm which employs a NFIC to realize IIC. The NFIC also chooses the long-term mean cell rate Λ_L and the short-term mean cell rate Λ_S as input variables and the increment T as the output variable; it adopts the same term sets for Λ_L, Λ_S, T and the same rule base as those employed by FIC, before the learning.

A. Structure of the NFIC

We adopt a five-layer neural fuzzy network [6], [10] to design the NFIC. Fig. 7 shows the structure of the NFIC. Nodes in layer one and layer five are input and output linguistic nodes, respectively. Nodes in layer two and layer four are term nodes which act as the membership functions for the respective input and output linguistic variables. Nodes in layer three are rule nodes, where each node represents one fuzzy rule and all nodes form a fuzzy rule base. Links in layer three and layer four function as an inference engine—layer-three links define preconditions and layer-four links define consequences of rule nodes. Links in layer two and layer five are fully connected between the linguistic nodes and their corresponding term nodes.

Generally, node i in layer k for the neural fuzzy controller has input function $f_i^{(k)}(u_{ij}^{(k)})$ and activation output function $a_i^{(k)}(f_i^{(k)})$, where $u_{ij}^{(k)}$ denotes the input to node i in layer k from node j in layer $(k-1)$ and $u_{ij}^{(k)}$ is expressed as $u_i^{(k)}$ if $k=1$. The layers for the NFIC are then designed as follows.

Layer 1: There are two input nodes for input linguistic variables Λ_L and Λ_S, and define

$$f_i^{(1)}\left(u_i^{(1)}\right)=u_i^{(1)}, \quad a_i^{(1)}=f_i^{(1)}, \quad 1\le i\le 2 \qquad (6)$$

where $u_1^{(1)}=\Lambda_L$ and $u_2^{(1)}=\Lambda_S$.

Layer 2: The nodes in this layer are used as the fuzzifier. As mentioned earlier, Λ_L and Λ_S have the same term set—$\{Low, Moderate, High\}$; thus, we have six nodes in the layer. Each node performs a bell-shaped function defined as

$$f_i^{(2)}\left(u_{ij}^{(2)}\right)=-\frac{\left(u_{ij}^{(2)}-m_{jn}^{(I)}\right)^2}{\left(\sigma_{jn}^{(I)}\right)^2}, \quad a_i^{(2)}=e^{f_i^{(2)}}, \quad 1\le i\le 6 \qquad (7)$$

where $u_{ij}^{(2)}=a_j^{(1)}$, $j=\lceil (i+2)/3\rceil$, and $m_{jn}^{(I)}$ and $\sigma_{jn}^{(I)}$ are the mean and the standard deviation of the nth term of the input linguistic variable from node j in layer 1, respectively, $n=i$ if $i\le 3$ and $n=i-3$ if $i>3$.

Layer 3: The links perform precondition matching of fuzzy control rules. There are nine rules in this layer as shown in Table I. Each rule node executes the fuzzy AND operation defined as

$$f_i^{(3)}\left(u_{ij}^{(3)}\right)=\min\left(u_{ij}^{(3)};\forall j\in P_i\right), \quad a_i^{(3)}=f_i^{(3)}, \quad 1\le i\le 9 \qquad (8)$$

where $u_{ij}^{(3)}=a_j^{(2)}$ and $P_i=\{j|\text{ all } j \text{ that are precondition nodes of the } i\text{th rule}\}$.

Layer 4: The nodes in this layer have two operating modes: $down-up$ and $up-down$. In the $down-up$ operating mode, the links perform consequence matching of fuzzy control rules. As shown in Fig. 5(b), there are five nodes in this layer. Each node performs a fuzzy OR operation to integrate the fired strength of rules that have the same consequence. Thus, we define

$$f_i^{(4)}\left(u_{ij}^{(4)}\right)=\max\left(u_{ij}^{(4)};\forall j\in C_i\right), \quad a_i^{(4)}=f_i^{(4)}, \quad 1\le i\le 5 \qquad (9)$$

where $u_{ij}^{(4)}=a_j^{(3)}$ and $C_i=\{j|\text{ all } j \text{ that have the same consequence of the } i\text{th term in the term set of } T\}$. In the $up-down$ operating mode which is used for training, the nodes in this layer

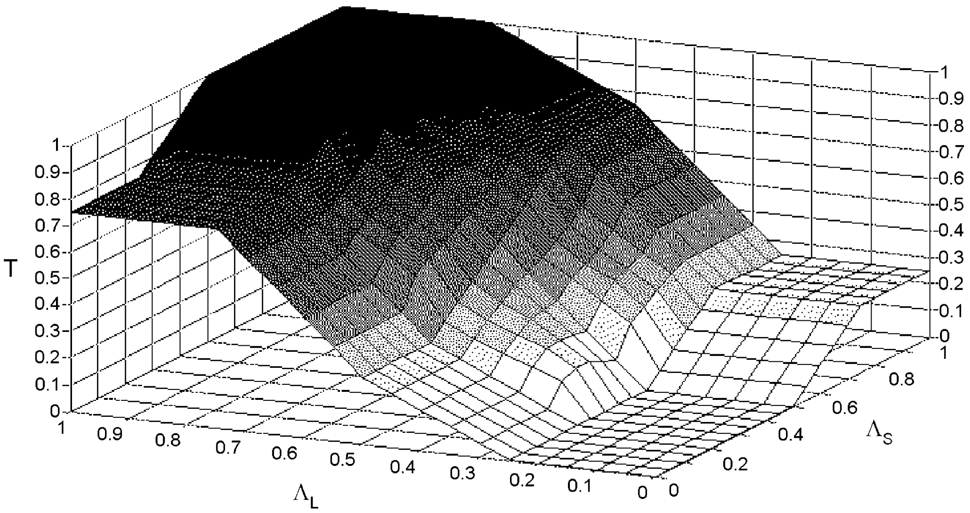


Fig. 6. Control surface of FIC.

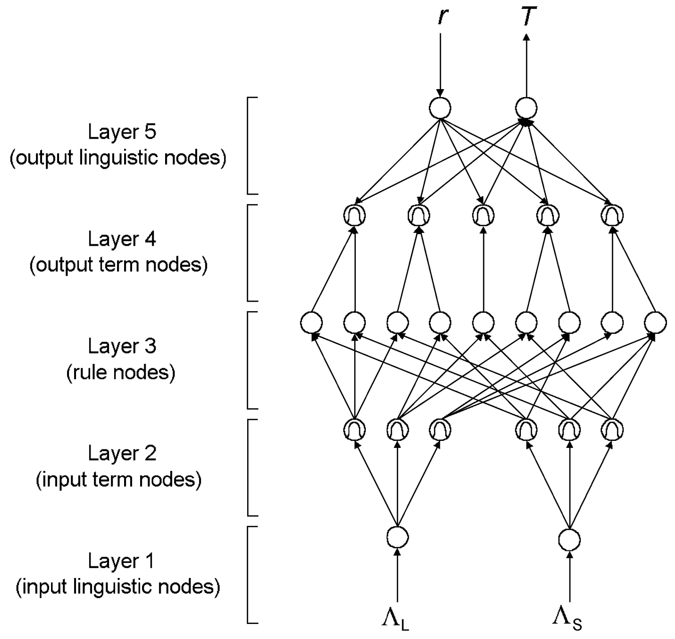


Fig. 7. Structure of the NFIC.

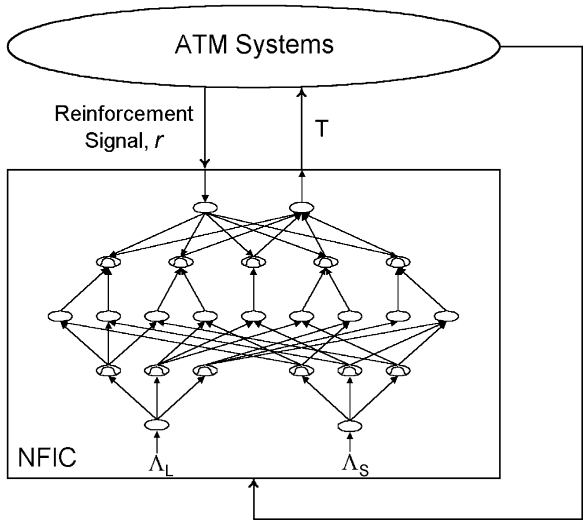


Fig. 8. Configuration of the reinforcement learning for the NFIC.

and the links in layer five have functions similar to those in layer two. Node i performs a bell-shaped function defined as

$$\tilde{f}_i^{(4)}\left(\tilde{u}_{ij}^{(4)}\right) = -\frac{\left(\tilde{u}_{ij}^{(4)} - m_i^{(O)}\right)^2}{\sigma_i^{(O)2}}, \quad \tilde{a}_i^{(4)} = e^{\tilde{f}_i^{(4)}}, \quad 1 \le i \le 5 \tag{10}$$

where "$\sim$" denotes the $up - down$ operation, $\tilde{u}_{ij}^{(4)} = \tilde{a}_j^{(5)}$ obtained from layer five, $j = 1$, and $m_i^{(O)}$ and $\sigma_i^{(O)}$ are the mean and the standard deviation of the ith term of the desired output.

Layer 5: There are two nodes in this layer. One node performs the $down - up$ operation for the actual output T. The node and its link act as the defuzzifier. The function used to simulate a *center of area* defuzzification method is approximated by

$$f_i^{(5)}\left(u_{ij}^{(5)}\right) = \sum_j \left(m_j^{(O)}\sigma_j^{(O)}\right) u_{ij}^{(5)}$$

$$a_i^{(5)} = \frac{f_i^{(5)}}{\sum_j \sigma_j^{(O)} u_{ij}^{(5)}}, \quad i = 1 \tag{11}$$

where $u_{ij}^{(5)} = a_j^{(4)}$, and $a_i^{(5)} = T$. The other node performs the $up - down$ operation during the training period. It should feed the desired output into the controller to adjust the link weight optimally. We set

$$\tilde{f}_i^{(5)}\left(\tilde{u}_i^{(5)}\right) = \tilde{u}_i^{(5)}, \quad \tilde{a}_i^{(5)} = \tilde{f}_i^{(5)}, \quad i = 1 \tag{12}$$

where $\tilde{u}_1^{(5)}$ is the desired output. Since we cannot obtain the desired output for the increment T, here we apply the *reinforcement learning* for the NFIC.

B. Reinforcement Learning

The diagram of the reinforcement learning for the NFIC is shown in Fig. 8, where the ATM system offers statistics of Λ_L and Λ_S input to the NFIC, provides a reinforcement signal r as a desired output to the NFIC, and receives the updating increment value T from the NFIC. The reinforcement signal is defined as

$$r = P_d^* - P_d \tag{13}$$

where P_d^* denotes the desired cell-loss ratio and P_d is the actual measured cell-loss ratio.

Based on this connectionist structure established in Fig. 7, the reinforcement learning is applied to optimally adjust parameters of input and output membership functions, according to the input training data, the reinforcement signal, the fuzzy partition, and the fuzzy logic rules. It derives updating rules for the mean and the standard deviation of the bell-shaped membership functions so as to minimize the error function, defined as

$$E = \frac{1}{2}r^2 = \frac{1}{2}\left(P_d^* - P_d\right)^2. \tag{14}$$

For each training data set, starting at the input nodes, the $down-up$ operation can compute to obtain the actual output of increment T. In the opposite direction, starting at the output node, the $up - down$ operation is used to compute $\partial E/\partial w$ for all hidden nodes, where w is the adjustable parameters such as the mean and the standard deviation for the input and output bell-shaped membership functions. We adopt the general learning rule to do the adjustment, which is given by

$$w(n+1) = w(n) + \eta \cdot \left(-\frac{\partial E}{\partial w}\right) \tag{15}$$

where η is the learning rate. The updating rules for parameters are layer by layer listed as follows.

Layer 5: The updating rule for $m_j^{(O)}$ in this layer can be obtained by

$$m_j^{(O)}(n+1) = m_j^{(O)}(n) + \eta \cdot r \cdot \frac{\sigma_j^{(O)} u_{ij}^{(5)}}{\sum_j \sigma_j^{(O)} u_{ij}^{(5)}}, \quad 1 \le j \le 5 \tag{16}$$

and the updating rule for $\sigma_j^{(O)}$ is given by

$$\sigma_j^{(O)}(n+1) = \sigma_j^{(O)}(n) + \eta \cdot r \cdot \frac{m_j^{(O)} u_{ij}^{(5)} \left(\sum_j \sigma_j^{(O)} u_{ij}^{(5)}\right) - u_{ij}^{(5)} \left(\sum_j m_j^{(O)} \sigma_j^{(O)} u_{ij}^{(5)}\right)}{\left(\sum_j \sigma_j^{(O)} u_{ij}^{(5)}\right)^2}. \tag{17}$$

The error signal $\delta^{(5)}$, to be propagated to the proceeding layer, is given by

$$\delta^{(5)} = r. \tag{18}$$

Layer 4: In this layer, only the error signal $\delta_i^{(4)}$ needs to be computed and propagated. $\delta_i^{(4)}$ is derived as

$$\delta_i^{(4)} = r \cdot \frac{m_i^{(O)} \sigma_i^{(O)} \left(\sum_i \sigma_i^{(O)} u_i^{(5)}\right) - \sigma_i^{(O)} \left(\sum_i m_i^{(O)} \sigma_i^{(O)} u_i^{(5)}\right)}{\left(\sum_i \sigma_i^{(O)} u_i^{(5)}\right)^2}. \tag{19}$$

Layer 3: As in layer 4, only the error signal, $\delta_i^{(3)}$, needs to be computed as

$$\delta_i^{(3)} = \delta_i^{(4)}. \tag{20}$$

Layer 2: The adaptive rule of $m_{ij}^{(I)}$ is derived as

$$m_{ij}^{(I)}(n+1) = m_{ij}^{(I)}(n) + \eta \cdot \delta_i^{(2)} \cdot e^{f_i^{(2)}} \cdot \frac{2\left(u_i^{(2)} - m_{ij}^{(I)}\right)}{\sigma_{ij}^{(I)2}} \tag{21}$$

and the adaptive rule of $\sigma_{ij}^{(I)}$ becomes

$$\sigma_{ij}^{(I)}(n+1) = \sigma_{ij}^{(I)}(n) + \eta \cdot \delta_i^{(2)} \cdot e^{f_i^{(2)}} \cdot \frac{2\left(u_i^{(2)} - m_{ij}^{(I)2}\right)}{\sigma_{ij}^{(I)3}} \tag{22}$$

where $\delta_i^{(2)} = -\sum_k q_k$; and $q_k = -\delta_k^{(3)}$ if $a_i^{(2)}$ is minimum in kth rule node's inputs; $q_k = 0$ otherwise.

V. Simulation Results

We verify the effectiveness of the intelligent leaky bucket algorithms in TS-UPC by comparing to the conventional leaky bucket algorithm. In the simulations, a two-state Markov modulated deterministic process (MMDP) source model, a two-state Markov modulated Bernoulli process (MMBP) source model, and a VBR MPEG video "Star Wars" are performed. We set the two-state MMDP and the two-state MMBP sources to have the mean active duration of 350 ms, the mean silence duration of 650 ms, and the mean cell rate $\Lambda_{mean} = 21.875$ cells/s. The holding time of each state of these two source models follows a geometric distribution. During the active state, the two-state MMDP source is a deterministic process which transmits cells at a fixed packetization interval of $T_{PCR} = 16$ ms, whereas the two-state MMBP source is a Bernoulli process which, for every fixed time interval $T_{PCR} = 1.6$ ms, is likely to transmit a cell with probability of 0.1. The PCR of the VBR MPEG video is 4000 cells/s, and the mean cell rate $\Lambda_{mean} = 975$ cells/s. The window size for calculating the short-term mean rate is set to be ten times the sum of the mean active duration and mean silent duration, i.e., window size $= 10 * (350 + 650)$ ms $= 10$ s.

In the simulations, C is set to be 1.1, thus for MMDP and MMBP sources, $\Lambda_{SCR} = C * \Lambda_{mean} = 24.0625$ cells/s. The increment T for the conventional leaky bucket algorithm, which is taken to be the inverse of the SCR, equals 0.041558, and the threshold τ_{SCR} of the leaky bucket equals $\tau_{IBT} + \tau'_{SCR}$, where $\tau_{IBT} = \lceil(\text{MBS} - 1)(T_{SCR} - T_{PCR})\rceil$ and $\tau'_{SCR} = T_{SCR}$ for MMDP and MMBP. In order to compare the performance under the MMDP and MMBP sources, τ_{SCR} for the MMBP source is set to be the same as the MMDP source. To calculate τ_{IBT} for the MMDP source, we need the MBS of the source. We set the allowed MBS for the MMDP source to be ten times the mean number of cell arrivals during the active state, i.e., MBS $= 10 * (350/16) = 218.75$ cells. Then τ_{SCR} can be calculated as 5.607. For the VBR MPEG video source, τ_{SCR} is set to be 3.79, and MBS $= 4870$ cells. For simplicity of simulation and not to distract our attention, the queue in TS is assumed to be of infinite capacity.

In this paper, we define Source σ as the ratio of the actual mean cell rate to the SCR of the traffic source. There are three

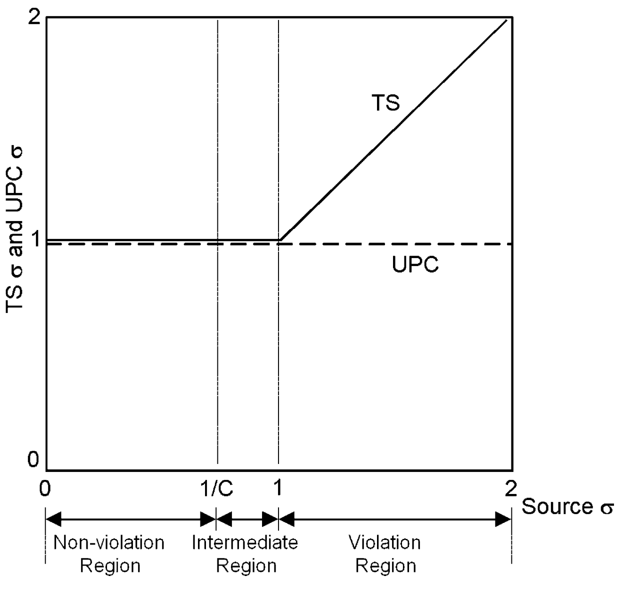


Fig. 9. Correspondence between TS σ, UPC σ, and Source σ.

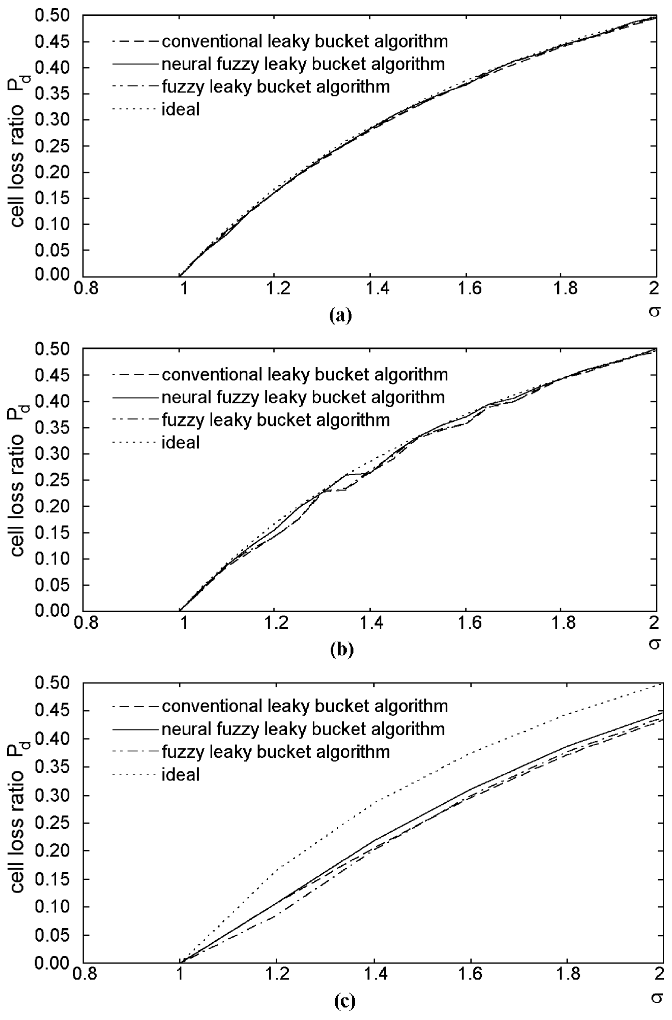


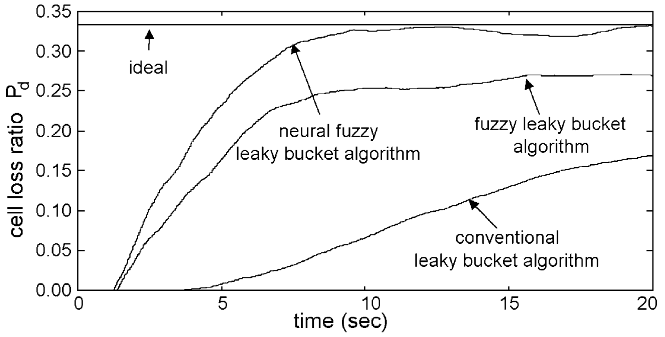
Fig. 10. Selectivity of the conventional leaky bucket algorithm, the fuzzy leaky bucket algorithm, and the neural fuzzy leaky bucket algorithm under: (a) MMDP traffic source; (b) MMBP traffic source; and (c) MPEG video traffic source.

regions for Source σ: *nonviolation region*, *intermediate region*, and *violation region*. The nonviolation region ranges from Source $\sigma = 0$ to Source $\sigma = 1/C$, where C is the magnifying factor. The user within this region is a legal user and is guaranteed a zero cell dropping (or tagging) probability imposed by UPC and a negligible queueing delay introduced by TS. The intermediate region is the region between Source $\sigma = 1/C$ and Source $\sigma = 1$. Any user within this region is also a legal user and can still have zero cell dropping probability, but it does not have a satisfactory queueing delay. Finally, the violation region is from those beyond Source $\sigma = 1$. The user whose Source σ is located in this region is an illegal user, and both the cell dropping probability and queueing delay are not guaranteed by TS-UPC.

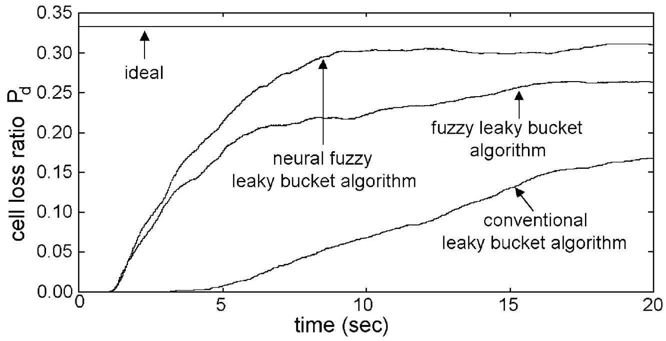
A connection with Source σ may have corresponding TS σ and UPC σ for TS and UPC, respectively, where TS σ $(\text{UPC } \sigma)$ is defined as the ratio of the allowed mean cell rate to the SCR at TS (UPC). As can be seen from Fig. 9, UPC σ is always held at 1 for all Source σ's such that UPC can pass legal cells but drop (tag) illegal cells. TS σ is fixed at 1 for Source $\sigma \leq 1$, denoting that legal cells can pass TS transparently; TS $\sigma =$ Source σ for Source $\sigma > 1$, denoting that the illegal calls can still pass TS in the sense that the badly-behaved user of the connection enlarges TS σ and intends to illegally enjoy a higher bit-rate service. If the user had not changed TS σ, then the cell stream passed by TS would have been conforming, even though Source $\sigma > 1$, but there would be tremendous queueing delay incurred.

Fig. 10(a)–(c) shows the selectivity for the conventional leaky bucket algorithm, the fuzzy leaky bucket algorithm, and the neural fuzzy leaky bucket algorithm, under the two-state MMDP traffic source, the two-state MMBP traffic source, and the MPEG video traffic source, respectively. The ideal curve of cell-loss ratio is $P_d^* = 1 - 1/(\text{Source } \sigma)$ for Source $\sigma > 1$ and $P_d^* = 0$ for Source $\sigma \leq 1$. As can be seen, the three algorithms present a zero cell-loss ratio for Source $\sigma \leq 1$. But for Source $\sigma > 1$, the neural fuzzy leaky bucket algorithm has a cell-loss ratio closest to the ideal curve, then the fuzzy leaky bucket algorithm and the conventional leaky bucket algorithm. In the case of the MPEG video traffic source, the difference phenomenon is more significant. It is because MPEG video traffic is burstier than MMDP and MMBP traffic sources and the intelligent TS-UPCs can be more adaptive than the conventional TS-UPC in dynamic, nonstationary systems.

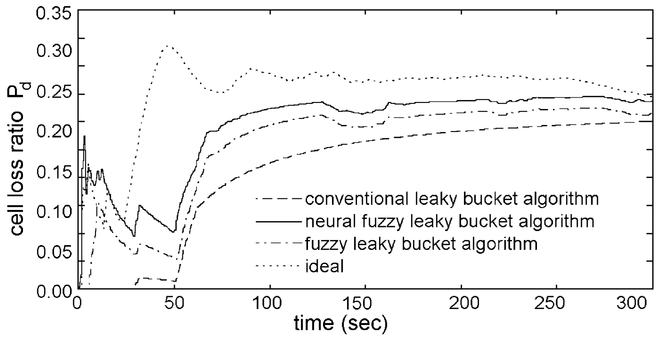
Fig. 11(a)–(c) shows the responsiveness behavior of the three leaky bucket algorithms under the two-state MMDP traffic source, the two-state MMBP traffic source, and the MPEG video traffic source, for Source $\sigma = 1.5$. The responsiveness is illustrated in terms of the cell-loss ratio versus time. From the figures, it can be seen that the intelligent leaky bucket algorithms not only have a shorter response time (i.e., the time it takes control action to start dropping the cells of a violating connection) which is about 1.5 s, as compared to 4 s of the conventional leaky bucket algorithm, but also has a higher detection rate (i.e., the rate the cell-loss ratio grows) than the conventional leaky bucket algorithm, under the MMDP, the MMBP, and the MPEG video traffic sources. It is because the adopted intelligent techniques have the ability to quickly express the control structure system using *a priori* knowledge; they are less dependent on the availability of a precise model



(a)

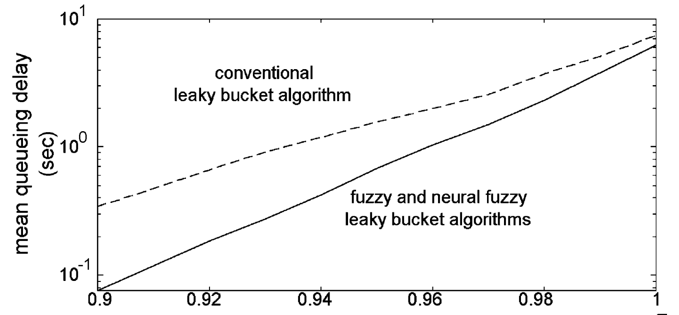


(b)

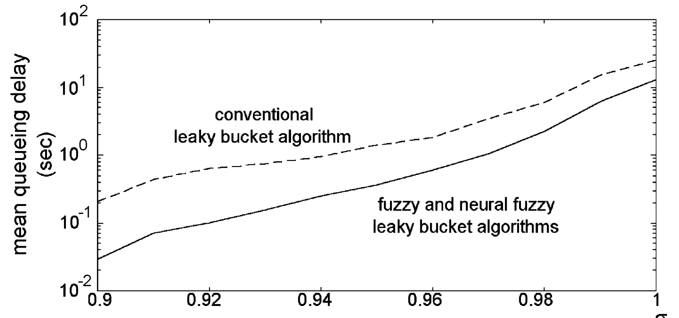


(c)

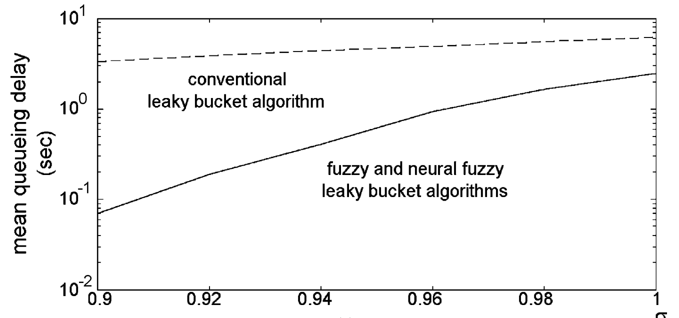
Fig. 11. Responsiveness of the conventional leaky bucket algorithm, the fuzzy leaky bucket algorithm, and the neural fuzzy leaky bucket algorithm under: (a) MMDP traffic source; (b) MMBP traffic source; and (c) MPEG video traffic source, for Source $\sigma = 1.5$.



(a)



(b)



(c)

Fig. 12. Mean queueing delay of the conventional leaky bucket algorithm, the fuzzy leaky bucket algorithm, and the neural fuzzy leaky bucket algorithm under: (a) MMDP traffic source; (b) MMBP traffic source; and (c) MPEG video traffic source.

of the controlled process and are more capable of handing nonlinearities. Also, the neural fuzzy leaky bucket algorithm performs better than the fuzzy leaky bucket algorithm. It is because the neural fuzzy network is a neural network structured on the basis of fuzzy logics; it integrates intelligent learning and computation of neural networks into fuzzy logic systems. Note that the fuzzy and neural fuzzy leaky bucket algorithms have similar detection rate to the fuzzy policer proposed in [13], but the former two have much earlier response time than the latter.

Fig. 12 shows the mean queueing delay versus different Source σ's under the two-state MMDP, the two-state MMBP, and the MPEG video traffic sources. We only consider the queueing delay of a connection with Source $\sigma \leq 1$ because the mean queueing delay of a violating connection needs not to be concerned. The figure reveals that the intelligent leaky bucket algorithms have the queueing delay more satisfactory than the conventional leaky bucket algorithm, regardless of the traffic source model used. This improvement owes to the fact that the intelligent leaky bucket algorithms further consider two system parameters, namely, the long-term and short-term mean rates. With these two parameters, the intelligent leaky bucket algorithms can know that the connection is conforming, so they set the increment to be very small in order to reduce the probability of cells being stored in the queue and thus decrease the mean queueing delay. Besides, the neural fuzzy leaky bucket algorithm has almost the same mean queueing delay as the fuzzy leaky bucket algorithm. Apparently, since the traffic source is legal, almost nothing can be learned from the reinforcement learning by the neural fuzzy leaky bucket algorithm to improve its performance.

VI. Concluding Remarks

In this paper, we employ the well-known intelligent techniques, which are the fuzzy logic systems and neural fuzzy networks, to design two intelligent leaky bucket algorithms for sustainable-cell-rate UPC of multimedia transmission in ATM networks. The first algorithm we proposed is the fuzzy leaky bucket algorithm, which as the name implies, employs a FIC in conjunction with the conventional leaky bucket algorithm. The FIC monitors the long-term mean rate and the short-term mean rate of a connection and uses the fuzzification, inference rules and defuzzification to process them in order to derive the optimal increment value. The other intelligent leaky bucket algorithm we proposed is the neural fuzzy leaky bucket algorithm, which utilizes a NFIC to dynamically adjust the increment value. The NFIC is basically an FIC, except that it further employs a neural network to optimize its fuzzy logic system through the reinforcement learning.

Simulation results show that, regardless of the traffic sources chosen, both intelligent leaky bucket algorithms achieve better performance in terms of selectivity, responsiveness, and mean queueing delay as compared to the conventional leaky bucket algorithm. The performance gain of the intelligent algorithms is a result of employing fuzzy logic and neural fuzzy network, as well as taking the long-term and short-term mean rates as the feedback information. Based on the feedback information, both intelligent algorithms can adapt to the time-varying and nonstationary traffic, and thus enhance their performance.

Simulation results also show that one of the intelligent leaky bucket algorithms outperforms the other. Compared with the fuzzy leaky bucket algorithm, the neural fuzzy leaky bucket algorithm achieves better performance in all aspects. Although the fuzzy logic is excellent in dealing with real-world impression and is capable of adapting itself to dynamic and bursty environments, it lacks the capability of automatically constructing its rule structure and membership functions to achieve the optimal performance. On the other hand, the neural fuzzy leaky bucket algorithm has perfected the impairment of the fuzzy leaky bucket algorithm by utilizing the learning capability of the neural network to continuously update the membership functions of the fuzzy logic system. However, the implementation cost of the neural fuzzy leaky bucket algorithm could be higher than that of the fuzzy leaky bucket algorithm.

Acknowledgment

The authors would like to thank the anonymous reviewers for their suggestions in improving the presentation of the paper.

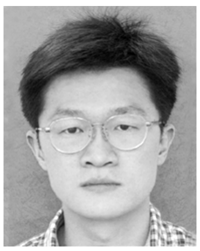
References

[1] ITU-T, Recommendation I.371, Geneva, May 1996.
[2] E. P. Rathgeb, "Modeling and performance comparison of policing mechanism for ATM networks," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 325–334, Apr. 1991.
[3] L. Dittmann, S. B. Jacobsen, and K. Moth, "Flow enforcement algorithms for ATM networks," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 343–350, Apr. 1991.
[4] S. Shioda and H. Saito, "Satisfying QoS standard with combined strategy for CAC and UPC," in *Proc. ICC '95*, pp. 965–969.
[5] M. Butto, E. Cavallero, and A. Tonietti, "Effectiveness of the leaky bucket policing mechanism in ATM networks," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 335–342, Apr. 1991.
[6] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems*. New York: Prentice-Hall, 1996.
[7] H. J. Ziemmermann, *Fuzzy Set Theory and Its Applications*, 2nd ed. New York: Kluwer, 1991, pp. 11–17.
[8] R. G. Cheng and C. J. Chang, "Design of a fuzzy traffic controller for ATM networks," *IEEE/ACM Trans. Networking*, vol. 4, pp. 460–469, June 1996.
[9] A. B. Bonde and S. Ghosh, "A comparative study of fuzzy versus "fixed" thresholds for robust queue management in cell-switching networks," *IEEE/ACM Trans. Networking*, vol. 2, no. 4, pp. 337–344, Aug. 1994.
[10] R. G. Cheng, C. J. Chang, and L. F. Lin, "A QoS-provisioning neural fuzzy connection admission controller for multimedia high-speed networks," *IEEE/ACM Trans. Networking*, vol. 7, pp. 111–121, Feb. 1999.
[11] A. A. Tarraf, I. W. Habib, and T. N. Saadwi, "A novel neural network traffic enforcement mechanism for ATM networks," *IEEE J. Select. Areas Commun.*, vol. 12, pp. 1088–1096, Aug. 1994.
[12] T. D. Ndousse, "Fuzzy neural control of voice cells in ATM networks," *IEEE J. Select. Areas Commun.*, vol. 12, no. 3, pp. 1488–1494, Dec. 1994.
[13] V. Catania, G. Ficili, S. Palazzo, and D. Panno, "A comparative analysis of fuzzy versus conventional policing mechanisms for ATM networks," *IEEE/ACM Trans. Networking*, vol. 4, pp. 449–459, June 1996.
[14] R. G. Garroppo, S. Giordano, S. Miduri, and F. Russo, "A prediction based UPC mechanism for VBR video traffic," in *Proc. ICC '98*, pp. 1119–1123.
[15] F. Guillemin, C. Rosenberg, and J. Mignault, "On characterizing an ATM source via the sustainable cell rate traffic descriptor," in *Proc. INFOCOM '95*, pp. 1129–1136.
[16] C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller—part I," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, pp. 419–435, Mar./Apr. 1990.

Chung-Ju Chang (S'81–M'85–SM'94) was born in Taiwan, R.O.C., in 1950. He received the B.E. and M.E. degrees in electronics engineering from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1972 and 1976, respectively, and the Ph.D. degree in electrical engineering from National Taiwan University (NTU), Taipei, in 1985.

From 1976 to 1988, he was with Telecommunication Laboratories, Directorate General of Telecommunications, Ministry of Communications, R.O.C., as a Design Engineer, Supervisor, Project Manager, and then Division Director. There, he was involved in designing digital switching systems, the RAX trunk tester, the ISDN user-network interface, and ISDN service and technology trials at the Science-Based Industrial Park. He also acted as a Science and Technical Advisor for the Minister of the Ministry of Communications from 1987 to 1989. In August 1988, he joined the faculty of the Department of Communication Engineering and Center for Telecommunications Research, College of Electrical Engineering and Computer Science, NCTU, as an Associate Professor. He has been a Professor since 1993. He was Director of the Institute of Communication Engineering from August 1993 to July 1995 and Chairman of Department of Communication Engineering from August 1999 to July 2001. He is currently the Dean of the Research and Development Office at NCTU. He has served as an Advisor for the Ministry of Education to promote the education of communication science and technologies for colleges and universities in Taiwan since 1995. He is also acting as a Committee Member for the Telecommunication Deliberate Body. His research interests include performance evaluation, wireless communication networks, and broadband networks.

Dr. Chang is a member of the Chinese Institute of Engineers (CIE).

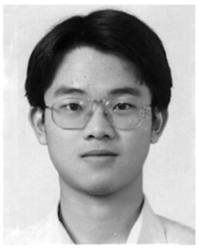


Chung-Hsun Yu received the B.E. and M.E. degrees in communication engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1996 and 1998, respectively.

Since 2001, he has been an Engineer with ZyXEL Communications Corporation, where he is involved in the designing of the virtual private network (VPN) on the Internet.

Chih-Sheng Chang received the B.E. and M.E. degrees in communication engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1997 and 1999, respectively.

Since 1999, he has been an Associate Researcher in the Broadband Transport and Access Technology Laboratory, Telecommunication Laboratories, Taiwan, where he is involved in VoIP and voice-packetized technology studies and the development of VoIP-related value-added services.



Li-Fong Lin (S'03) received the B.E. degree in communication engineering in 1996 from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, R.O.C., where he is currently pursuing the Ph.D. degree in the area of communication engineering.

His research interests include performance analysis, traffic control over high-speed multimedia networks, fuzzy systems, and neural networks.