# A Modified Random Early Detection Algorithm: Fuzzy Logic Based Approach

Mohammad Hossein Yaghmaee

*Abstract:* **In this paper, a fuzzy logic implementation of the random early detection (RED) mechanism [1] is presented. The main objective of the proposed fuzzy controller is to reduce the loss probability of the RED mechanism without any change in channel utilization. Based on previous studies, it is clear that the performance of RED algorithm is extremely related to the traffic load as well as to its parameters setting. Using fuzzy logic capabilities, we try to dynamically tune the loss probability of the RED gateway. To achieve this goal, a two-input-single-output fuzzy controller is used. To achieve a low packet loss probability, the proposed fuzzy controller is responsible to control the $\max_p$ parameter of the RED gateway. The inputs of the proposed fuzzy controller are 1) the difference between average queue size and a target point, and 2) the difference between the estimated value of incoming data rate and the target link capacity. To evaluate the performance of the proposed fuzzy mechanism, several trials with file transfer protocol (FTP) and burst traffic were performed. In this study, the ns-2 simulator [2] has been used to generate the experimental data. All simulation results indicate that the proposed fuzzy mechanism outperforms remarkably both the traditional RED and Adaptive RED (ARED) mechanisms [3]–[5].**

*Index Terms:* **Active queue management (AQM), congestion control, fuzzy logic control, random early detection (RED).**

## I. INTRODUCTION

The number of Internet users is rapidly growing. By increasing the number of users, the amount of traffic is also increased. When arrival rate to a router is greater than its departure rate, congestion can be occurred. To prevent the computer network from becoming a bottleneck, traffic management is necessary. In circuit switch networks, i.e., public switch telephone network (PSTN), each connection uses a fixed amount of bandwidth and data is transmitted at a constant bit rate. So, call acceptance procedure is very simple in circuit switch networks. In packet switch networks due to random nature of input traffic and fault condition within the network, the traffic control function is a very complex task. The Internet is increasingly facing packet loss and queuing delays due to its rapid growth. This can lead to congestion collapse, which will reduce the quality of Internet applications [6]. It is important for the current Internet to support various traffic classes with variable bit rates. To support new Internet applications such as voice over Internet protocol (IP), video on demand, multimedia, and electronic commerce, it is necessary to design of effective congestion control and queue management algorithms. However, such a design is known to be difficult, because of variety of services supported in the Internet and their various demands for quality of service (QoS). As the most current Internet traffic is bursty, routers are provisioned with fairly large buffers to absorb this burstiness and maintain high link utilization. Active queue management (AQM) techniques try to detect and react to the congestion before its consequences such as packet loss or queuing delays. In reaction to suspected congestion, AQM algorithms drop packets early or explicit congestion notification (ECN)-mark them to inform the congestion to the traffic sources. The most important difference among AQM schemes is that when they guess congestion and how do they select the packets to be marked/dropped. AQM is the pro-active approach of informing the sender about incipient congestion before a buffer overflow happens. By using AQM mechanisms, the senders are informed early about congestion and can react accordingly. Random early detection (RED) [1] is the most important AQM mechanism which was proposed in order to solve problems caused by drop tail (DT) queue management mechanism. RED uses randomization to solve both the lockout and full queue problems in an efficient manner without requiring any changes at the end hosts. RED simply sets minimum and maximum dropping thresholds. If the average buffer size exceeds the minimum threshold, RED starts randomly dropping packets based on a probability depending on the queue length. If buffer size exceeds the maximum threshold, then every packet is dropped. As expressed completely in [7] and [8], RED contains severe problems. The fundamental one is that it uses queue length as a congestion indicator. This indicator cannot completely show the severity of congestion. On the other hand, average queue length varies with the level of congestion and with the parameter settings. As a result, the performance of RED is too sensitive to traffic load and parameter settings [8]. That is, when the link is lightly congested and/or $\max_p$ is high, the average queue size is near $\min_{th}$; when the link is more heavily congested and/or $\max_p$ is low, the average queue size is closer to or even above $\max_{th}$. Different variants of RED such as stabilized RED (SRED) [9] and adaptive RED (ARED) [3]–[5] have been proposed which could fix some of its shortcomings. In [10]–[12], some AQM mechanisms were proposed which use flow based congestion indicator. To show the sensitivity of RED algorithm to the parameter, two different values 1 and 0.01 are considered for $\max_p$. In the following sections, RED1 and RED0.01 refer the RED algorithm with $\max_p = 1$ and $\max_p = 0.01$, respectively.

During the past few years, fuzzy logic has been found many applications in telecommunication networks [13]. In [14]–[25], many different fuzzy logic controllers were proposed for traffic management in the asynchronous transfer mode (ATM) net-

Manuscript received December 18, 2003; approved for publication by Suresh Subramaniam, Division III Editor, April 6, 2005.

The author is with the Institute for Studies in Theoretical Physics and Mathematics (I.P.M.), Computer Department, Ferdowsi University of Mashhad, Iran, email: hyaghmae@ferdowsi.um.ac.ir.

works including call admission control (CAC) and usage parameter control (UPC). The concept of fuzzy threshold and adaptive buffer management was proposed in [26]. As shown in this paper, the fuzzy threshold buffer management has a better performance than the traditional binary thresholds. In [27], a framework that offers the QoS in a differentiated services (DiffServ) domain using both policy-based management and fuzzy logic techniques was proposed. Furthermore, the performance and functionalities of the proposed model were shown by simulation of a voice over IP application in different DiffServ topologies. In [28], the fuzzy explicit marking (FEM) was proposed, which can support explicit congestion notification (ECN), to provide congestion control in TCP/IP best-effort networks using a fuzzy logic control approach. In [29], a methodology was proposed to choose optimized fuzzy controller parameters using the Wang-Mendel and genetic algorithms and was simulated for voice over IP applications in DiffServ domains. In [30], a fuzzy logic approach for RED implementation was developed for DiffServ.

Based on previous studies, the dynamic tuning of $\max_p$ parameter of RED algorithm give us better performance. In [3] and [4], Feng proposed the original ARED mechanism. This mechanism retains RED's basic structure and merely adjusts the parameter $\max_p$ to keep the average queue size between $\min_{th}$ and $\max_{th}$. In [4], some simulation results illustrates RED's sensitivity to parameters, and shows that ARED does indeed address this problem. Based on results shown in [4], the RED's average queue size and its performance vary as a function of the RED parameters $\max_p$. Furthermore, it is shown that by adapting $\max_p$ to keep the target queue size within a target range between $\min_{th}$ and $\max_{th}$ it is possible to achieve the good performance.

This paper introduces a fuzzy logic based control design to reduce the loss probability of the RED mechanism. The main objective of the proposed model is to tune the loss probability of the RED mechanism so that its average queue size remains nearly constant. The proposed fuzzy controller has two inputs and a single output. The inputs of fuzzy controller are

1. the error signal $e_1$, which is calculated as the difference between average queue size ($avg$) and a target point, $TARGET$,
2. the error signal $e_2$, which is calculated as the difference between the estimated incoming data rate $C_{est}$ and the target's link capacity $C_t$.

The output of the proposed fuzzy controller is used to calculate a new value of $\max_p$. The main objective the fuzzy controller is to control the average queue size near a target point. When $avg$ is less than $TARGET$ and the incoming data rate is less than $C_t$, $\max_p$ is decreased which in turn decreases the loss rate. On the other hand, when both $avg$ is greater than $TARGET$ and $C_{est}$ is greater than $C_t$, $\max_p$ is increased; this will increase the loss rate. By controlling $\max_p$ dynamically, the proposed fuzzy mechanism will achieve a low loss rate.

The remainder of this paper is organized as follows. Section II describes fully the proposed fuzzy logic controller. In Section III, by using computer simulation, the performance of the fuzzy controller is compared with that of the original RED algorithm. Two adaptive RED algorithms, are explained in Section IV. Furthermore, in this section the performance of the proposed model is compared with that of the ARED mechanism. Section V is devoted to present the performance of proposed fuzzy controller for DiffServ. Finally, Section VI concludes the paper.

## II. THE PROPOSED FUZZY CONTROLLER

In this section, the proposed fuzzy controller is explained in detail. For this purpose, first a brief introduction to the fuzzy logic controller and its concepts are presented, then the proposed fuzzy logic based approach is explained in details.

### A. Overview of Fuzzy Logic Controllers (FLC)

The use of fuzzy logic is rapidly spreading in the realm of consumer products design in order to develop control systems with nonlinear characteristics. Fuzzy logic controllers, like expert systems, are used to model human experience and human decision making behavior. In FLC, the input-output relationship is expressed by using a set of linguistic rules or relational expressions. An FLC basically consists of four important parts including a fuzzifier, a defuzzifier, an inference engine, and a rule base [31]. As in many fuzzy control applications, the input data are usually crisp, so a fuzzification is necessary to convert the input crisp data into a suitable set of linguistic value which is needed in inference engine. The singleton fuzzifier, maps a real-valued point $X^*$ into a fuzzy singleton $A'$ which has membership value 1 at $X^*$ and 0 at all other points. The main advantage of using singleton fuzzifier is the great simplicity of implementing the consequence part. It can be used with Mamdani' method to simplify considerably the defuzzification stage, whose task is reduced to the calculation of a weighted average with a restricted set of crisp values. The use of singletons has no bad consequence on the output variable domain which can be the same as with triangular or trapezoid output sets when using the center of gravity defuzzification method. In the rule base of an FLC, a set of fuzzy control rules, which characterize the dynamic behavior of system, are defined. It is the heart of the fuzzy system in the sense that all other components are used to implement these rules in a reasonable and efficient manner. The inference engine is used to form inferences and draw conclusions from the fuzzy control rules. In a fuzzy inference engine, fuzzy logic principles are used to combine the fuzzy rules into a mapping from input fuzzy sets to the output fuzzy sets. There are a number of fuzzy inference engines that are commonly used in fuzzy systems and fuzzy control. The product and minimum inference engines are the most commonly inference engine techniques. The output of inference engine is sent to defuzzification unit. Defuzzification is a mapping from a space of fuzzy control actions into a space of crisp control actions. Conceptually, the task of the defuzzifier is to specify a point that best represents the output fuzzy set. The center of gravity, center average, and maximum (or hight) are the most commonly defuzzification techniques. The common center of gravity defuzzification method requires a quantity of calculation that is prohibitive for many real-time applications with software implementations. Its calculation can however be simplified when associated with the sum product method. The computation of the center of gravity can take advantage of the high speed afforded by VLSI when integrated on an IC, which

is however quite complex.

Suppose we have an FLC with $n$ inputs including $x_1, x_2, \cdots, x_n$ and one output $y \in R$. The input vector $X$ is defined as $X = (x_1, x_2, \ldots, x_n)^T \in R^n$. Furthermore, suppose the rule base consists of $M$ rules with the following general form:

Rule 1: If $x_1$ is $A_1^1$, $x_2$ is $A_2^1$, $\cdots$, and $x_n$ is $A_n^1$, then $y$ is $B^1$;
Rule 2: If $x_1$ is $A_1^2$, $x_2$ is $A_2^2$, $\cdots$, and $x_n$ is $A_n^2$, then $y$ is $B^2$;
$\vdots$
Rule $M$: If $x_1$ is $A_1^M$, $x_2$ is $A_2^M$, $\cdots$, and $x_n$ is $A_n^M$, then $y$ is $B^M$;

where in the $i$-th rule, $A_j^i$ and $B^i$ $(i = 1, 2, \cdots, M\ ;\ j = 1, 2, \cdots, n)$ are fuzzy sets of linguistic variable $x_j$ and $y$, respectively. In [31], it is shown that the output $f(x) \in R$ of this fuzzy controller with singleton fuzzifier, minimum inference engine, and center average defuzzifier is calculated as

$$f(x) = \frac{\sum_{i=1}^{M} \overline{y}^i (\min_{j=1}^{n} \mu_{A_j^i}(x_j))}{\sum_{i=1}^{M} (\min_{j=1}^{n} \mu_{A_j^i}(x_j))}$$

where $\overline{y}^i$ is the center of fuzzy set $B^i$ and $\mu_{A_j^i}(x_j)$ is the membership function of fuzzy set $A_j^i$ of linguistic variable $x_j$ in the $i$-th rule $(i = 1, 2, \cdots, M)$.

### *B. Proposed FLC Based Approach*

In this subsection, the proposed fuzzy logic approach for controlling the $\max_p$ parameter of RED algorithm is presented. Both the table look-up scheme and the trial and error approach are used. In the table look-up scheme, the fuzzy sets are defined to cover the input and output spaces. After defining the input and output fuzzy sets, a set of fuzzy rules are collected, then the fuzzy controller is constructed from these fuzzy rules; finally, the fuzzy controller is tested and if the performance is not satisfactory, the rules are fine-tuned or designed in a number of trial and error cycles until the performance is satisfactory. The design is summarized in the following steps:

**• Step 1: Analyze the RED algorithm and chosse state and control variables:**

As it mentioned earlier, the RED algorithm suffers from two main problems [5]. The first problem is that it uses the queue length as a congestion indicator that cannot completely show the severity of the congestion. The second problem is that it is very sensitive to its parameters' variations. To achieve a good throughput and reasonable average queue lengths, a fuzzy logic controller is used to dynamically tune the $\max_p$. As the measured queue length and measured incoming data rate vary, the proposed fuzzy controller dynamically tune the value of $\max_p$ in order to reduce the loss rate while keeping the channel utilization fixed. The fuzzy controller uses the following two input parameters:

1. The error signal $e_1$ which is calculated as $e_1 = avg - TARGET$, where $avg$ is the average buffer size and $TARGET$ is the target queue size determined by

$$TARGET = \frac{\max_{th} + \min_{th}}{2}.$$

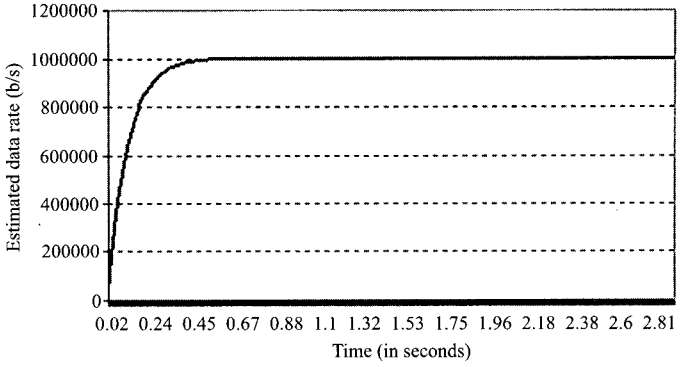


Fig. 1. The data rate estimation of a 1 Mb/s CBR source.

2. The error signal $e_2$ which is calculated as the difference between the estimated incoming data rate ($C_{est}$) and the target link capacity ($C_t$). Usually $C_t$ is set to $0.97C$, where $C$ is the link capacity. To estimate the incoming data rate, the exponential averaging formula given below is employed.

$$C_{est} = \frac{(1 - e^{\frac{-Del}{K}})B}{Del} + C_{est} e^{\frac{-Del}{K}}.$$

In the above, $Del$ is the inter-packet delay, $B$ stands for the packet size, and $K$ is the time constant (usually $K$ is set to 0.9). To evaluate the accuracy of the data rate estimator, a simple simulation in ns-2 simulator was performed. The rate estimator was applied to calculate the data rate of a single 1 Mb/s CBR traffic source. Fig. 1 shows the results. As easily observed, the rate estimator determined the incoming data rate perfectly.

We believe that the proposed fuzzy controller is able to tackle the following pitfalls of the original RED algorithm.

1. RED uses the average queue size as a congestion indicator. The queue size is not a good indicator of the severity of the congestion, and the level of congestion notifications issued may be too bursty, leading to an excessive packet loss. The proposed fuzzy controller adjusts the rate of the congestion notification in response to a flow based congestion measure.
2. RED is very sensitive to both traffic load and its parameters settings. To remedy this, the proposed fuzzy controller is designed to successfully auto-tune the parameter to achieve reliably good results. To do so, the major goal of the fuzzy controller will be to make the average buffer size vary in the neighborhood of the target queue size.

**• Step2: Derive fuzzy sets to cover the input and output spaces:**

It is assumed that the error signals $e_1$ and $e_2$ are belonged to the interval $[\alpha_1, \beta_1], [\alpha_2, \beta_2]$, respectively. The values of $\alpha_1$, $\alpha_2$, $\beta_1$, and $\beta_2$ were set as below

$$\alpha_1 = \frac{\min_{th} - \max_{th}}{2}\ ,\ \beta_1 = \frac{\max_{th} - \min_{th}}{2},$$

$$\alpha_2 = -0.097C_t\ ,\ \beta_2 = 0.097C_t.$$

For each $[\alpha_1, \beta_1], [\alpha_2, \beta_2]$, define $N_i$ $(i = 1, 2)$ fuzzy sets $A_i^j$ $(j = 1, 2, \cdots, N_i)$, $(i = 1, 2)$ which are required to be complete in $[\alpha_i, \beta_i]$; that is, for any $x_i \in [\alpha_i, \beta_i]$, there exists $A_i^j$ such that $\mu_{A_i^j}(x_i) \neq 0$.

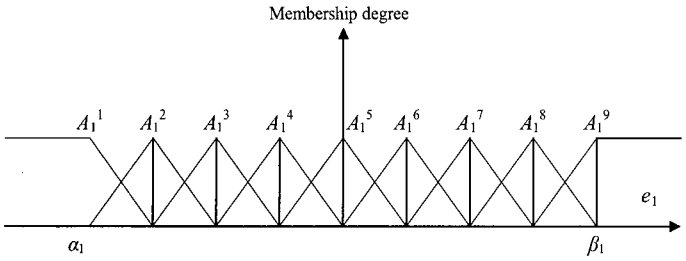


Fig. 2. The membership functions of input $e_1$.

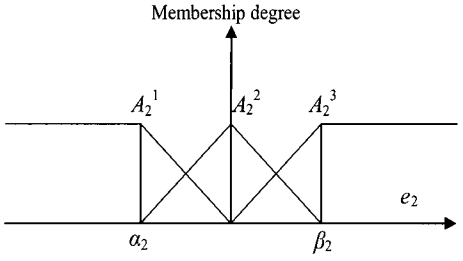


Fig. 3. The membership functions of input $e_2$.

Since triangular membership function offer more computational simplicity, the triangular function $\mu(x;a,b,c)$ which is defined as below is used

$$\mu(x;a,b,c)=\begin{cases}0, & \text{for } x>0 \text{ or } x>c\\ \dfrac{x-a}{b-a}, & \text{for } a\le x\le b\\ \dfrac{c-x}{c-b}, & \text{for } b\le x\le c.\end{cases}$$

The triangular membership functions

$$\mu_{A_i^j}(x_i)=\mu_{A_i^j}(x_i;a_i^j,b_i^j,c_i^j)$$

is chosen, where

$$\begin{aligned}&A_i^1<A_i^2<\cdots<A_i^{N_i},\\ &a_i^1=-\infty,\ b_i^1=\alpha_i,\ b_i^j=a_i^{j+1},\\ &b_i^{j+1}=c_i^j=a_i^{j+1}+\tfrac{\beta_i-\alpha_i}{N_i-1},\ (j=1,2,\cdots,N_i-1),\\ &b_i^{N_i}=\beta_i,\ c_i^{N_i}=\infty.\end{aligned}$$

Figs. 2 and 3 shows the inputs membership functions of the proposed fuzzy controller. As shown in Figs. 2 and 3, the values of $N_1$ and $N_2$ are equal to 9 and 3, respectively.

**• Step3: Construct $M=N_1\times N_2$ fuzzy IF-THEN rules that relate the state variables with the control variable:**

To construct the fuzzy rules, we use the following fuzzy IF-THEN rule:

$$Rule^{i_1i_2}: \text{IF } e_1 \text{ is } A_1^{i_1} \text{and } e_2 \text{ is } A_2^{i_2}, \text{THEN } y \text{ is } B^{i_1i_2}$$

where $i_1=1,2,\cdots,N_1$, $i_2=1,2,\cdots,N_2$. Note that in this case we need to define $M=9\times 3=27$ fuzzy sets for $B^{i_1i_2}$. To reduce the complexity of the fuzzy controller, only $N_y=9$ fuzzy sets $B^j$, $j=1,2,\cdots,N_y$, where are complete in $[\alpha_y,\beta_y]$ are defined. The values of $\alpha_y$ and $\beta_y$ were set to $\alpha_y=-0.3\max_p$ and $\beta_y=\min(0.008,0.25\max_p)$, respectively. We also used the triangular membership functions:

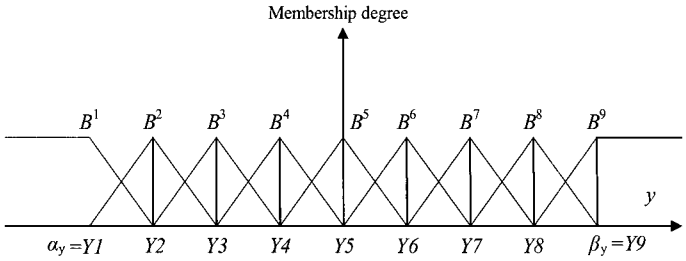


Fig. 4. The membership functions of the output $y$.

| $e_2$ \ $e_1$ | $A_1^1$ | $A_1^2$ | $A_1^3$ | $A_1^4$ | $A_1^5$ | $A_1^6$ | $A_1^7$ | $A_1^8$ | $A_1^9$ |
|---|---|---|---|---|---|---|---|---|---|
| $A_2^1$ | $B^1$ | $B^1$ | $B^2$ | $B^3$ | $B^4$ | $B^5$ | $B^5$ | $B^5$ | $B^6$ |
| $A_2^2$ | $B^1$ | $B^2$ | $B^3$ | $B^5$ | $B^5$ | $B^6$ | $B^7$ | $B^8$ | $B^9$ |
| $A_2^3$ | $B^2$ | $B^3$ | $B^4$ | $B^5$ | $B^6$ | $B^7$ | $B^8$ | $B^9$ | $B^9$ |

Fig. 5. Table look-up illustration of the fuzzy rule base of the proposed fuzzy controller.

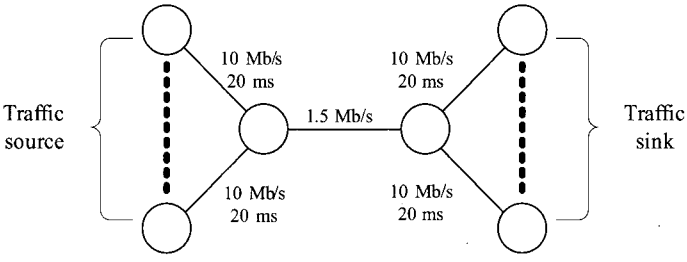


Fig. 6. The simulation topology.

$\mu_{B^j}(y)=\mu_{B^j}(y;a^j,b^j,c^j)$ where $B^1<B^2<\cdots<B^{N_y}$, $a^1=-\infty$, $b^1=\alpha_y$, $b^j=a^{j+1}<b^{j+1}=c^j(j=1,2,\cdots,N_y-1)$, and $b^{N_y}=\beta_y$, $c^{N_y}=\infty$.

Fig. 4 shows the membership functions of the output $y$. The values of $Y1$, $Y2$, $Y3$, $Y4$, $Y5$, $Y6$, $Y7$, $Y8$, and $Y9$ were set to $-0.3\max_p$, $-0.25\max_p$, $-0.15\max_p$, $-0.1\max_p$, 0, $\min(0.001,0.15\max_p)$, $\min(0.003,0.2\max_p)$, $\min(0.005,0.23\max_p)$, and $\min(0.008,0.25\max_p)$, respectively. Note that the above constant values were fine-tuned in a number of trial and error cycles until the loss performance of the proposed fuzzy controller is satisfactory.

Intuitively, we can illustrate the fuzzy rule base as a look-up table in the two input case. Fig. 5 shows a combination of fuzzy sets in $[\alpha_1,\beta_1]$, fuzzy sets in $[\alpha_2,\beta_2]$, and thus a possible rule.

According to the proposed fuzzy rule base, when both $avg$ and the estimated data rate are low ($e_1$ and $e_2$ are negative), $\max_p$ is decreased; this causes most of input packets to be passed through the router without any loss leading to a decrease in the loss probability. On the other hand, when $avg$ and the estimated data rate are both high ($e_1$ and $e_2$ are positive), $\max_p$ is increased which causes more of input packets to be lost. This in turn leads to an increase in loss probability.

**• Step4: Combine derived fuzzy rules into a fuzzy system and test the closed-loop system with this fuzzy system as controller:**

In the proposed fuzzy controller, singleton fuzzifier, mini-

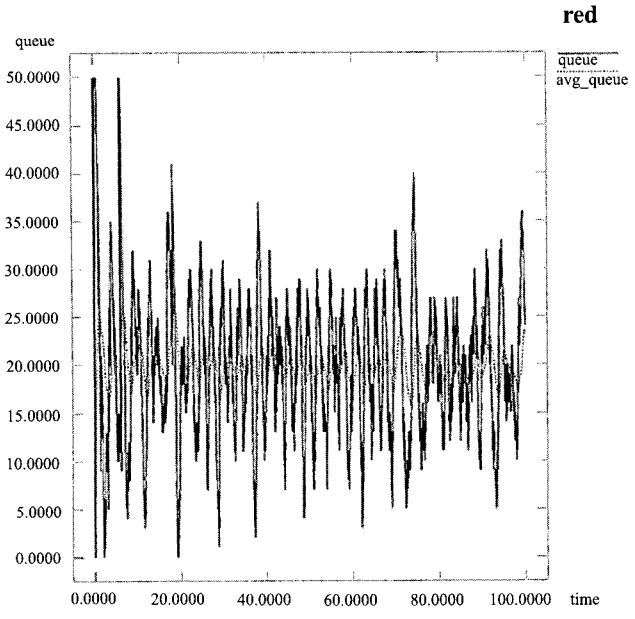


Fig. 7. The queue size of RED1.

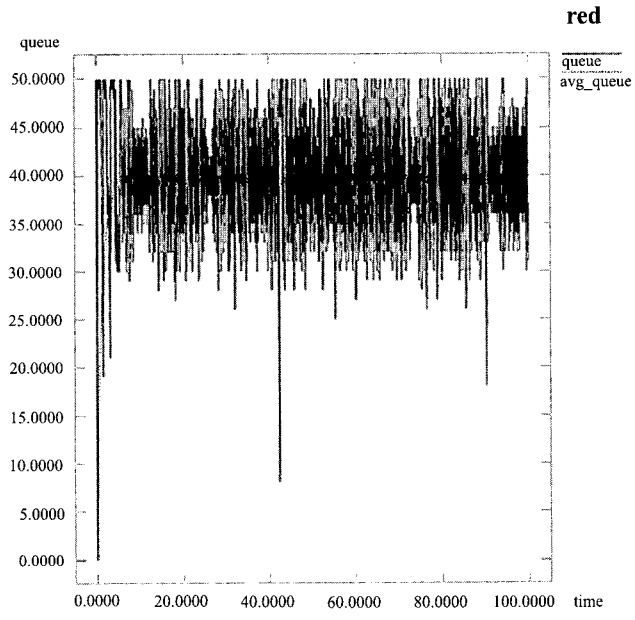


Fig. 8. The queue size of RED0.01.

mum inference engine, and center average defuzzifier are used. The current value of input linguistic variables $e_1$ and $e_2$ are mapped to their membership functions and the output of fuzzy logic controller, $delta\_\max\_p$, is calculated as

$$delta\_\max\_p = \frac{\sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2} \overline{y}^{i_1 i_2} \min\left\{\mu_{A_1^{i_1}}(e_1), \mu_{A_2^{i_2}}(e_2)\right\}}{\sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2} \min\left\{\mu_{A_1^{i_1}}(e_1), \mu_{A_2^{i_2}}(e_2)\right\}}$$

where $\overline{y}^{i_1 i_2}$ are the elements of matrix $Y_{N_1 \times N_2}$ which is defined

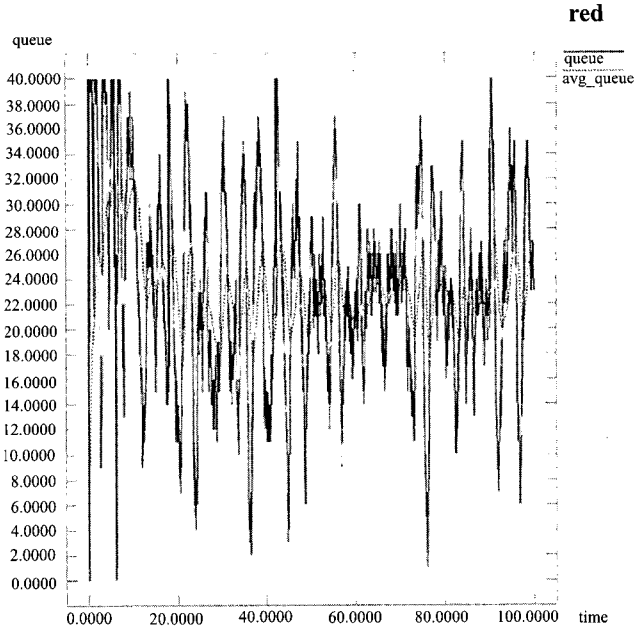


Fig. 9. The queue size of Fuzzy.

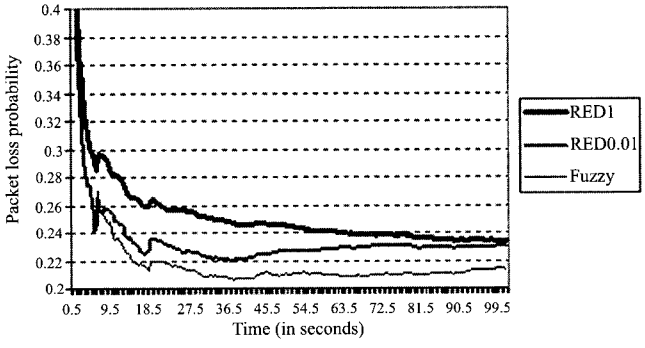


Fig. 10. Packet loss probability of RED1, RED0.01, and Fuzzy versus simulation time (at 100 FTP sources).

as below

$$Y_{N_1 \times N_2} = [\overline{y}^{i_1 i_2}]_{9\times 3}$$

$$= \begin{bmatrix} Y1 & Y1 & Y2 & Y3 & Y4 & Y4 & Y5 & Y5 & Y6 \\ Y1 & Y2 & Y3 & Y5 & Y5 & Y6 & Y7 & Y8 & Y9 \\ Y2 & Y3 & Y4 & Y5 & Y6 & Y7 & Y8 & Y9 & Y9 \end{bmatrix}^T.$$

The values of $Y1, Y2, \cdots, Y9$ were defined previously. Since the fuzzy sets $A_i^1$, $A_i^2$, $\cdots$, $A_i^{N_i}$ are complete, at every point $(e_1, e_2)$ there exist $i_1$ and $i_2$ such that $\min\left\{\mu_{A_1^{i_1}}(e_1), \mu_{A_2^{i_2}}(e_2)\right\} \neq 0$. Hence, the proposed fuzzy system is well defined, that is, its denominator is always nonzero. The output of fuzzy controller, $delta\_\max\_p$, demonstrates the change value in the $\max_p$. At the end of each periodic time interval $T$ (typically $T = 0.2$ sec.), the new value of $\max_p$ is calculated as

$$\max_p(nT) = \max_p((n-1)T) + delta\_\max\_p(nT), \quad n = 1, 2, \cdots.$$

Note that at the end of each time interval, the value of $\max_p$ is constant until the next time interval. It is clear that in the pro-

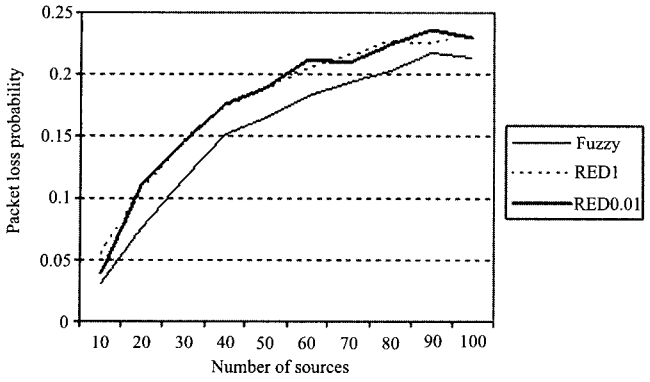


Fig. 11. Packet loss probability of RED1, RED0.01, and Fuzzy versus number of traffic sources (at 100 FTP sources).

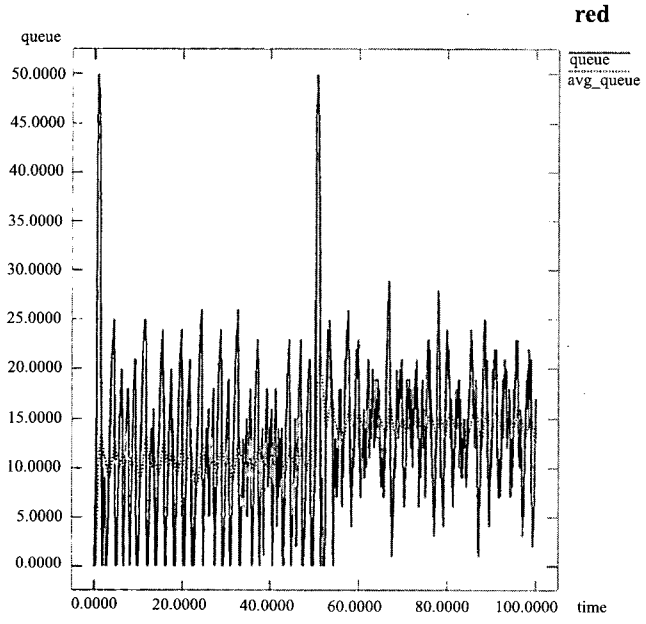


Fig. 12. RED1 with an increase in congestion.

posed fuzzy controller, the loss probability of the RED mechanism is dynamically tuned based on the level of congestion.

## III. SIMULATION

In this section, by using the ns-2 simulator, we compare the performance of the proposed fuzzy controller with that of traditional RED mechanism. For this purpose, the proposed fuzzy controller was added as a new queue management algorithm to the ns-2. From now on, we use the acronym of Fuzzy for the resulting queue management algorithm. As mentioned before, to show the sensitivity of RED algorithm to the $\max_p$ parameter, we consider two different values 1 and 0.01 for $\max_p$. In the following figures, RED1 and RED0.01 refer the RED algorithm with $\max_p = 1$ and $\max_p = 0.01$, respectively. The network topology used for simulation is a single congested link in a dumbbell topology shown in Fig. 6. As shown in this figure, some TCP traffic sources are directly connected to a network router. To evaluate the performance of the fuzzy controller, we consider both FTP and bursty traffic. For FTP traffic, all traffic sources always have a packet of size 1000-byte to send as soon

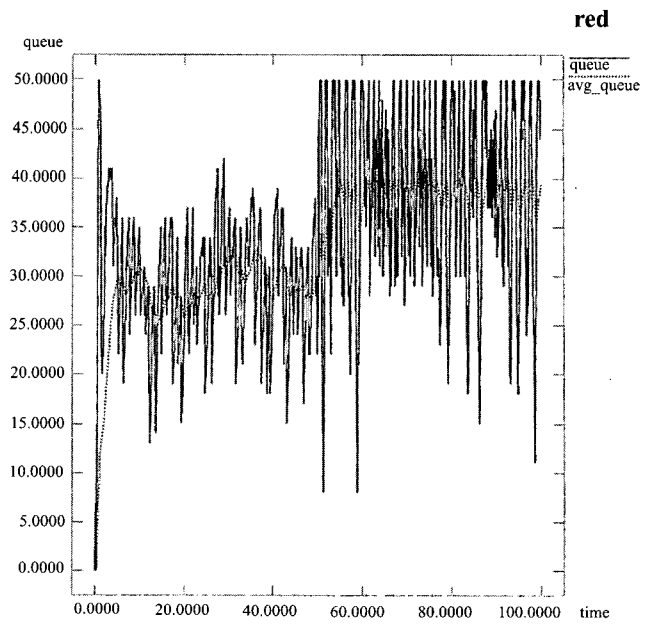


Fig. 13. RED0.01 with an increase in congestion.

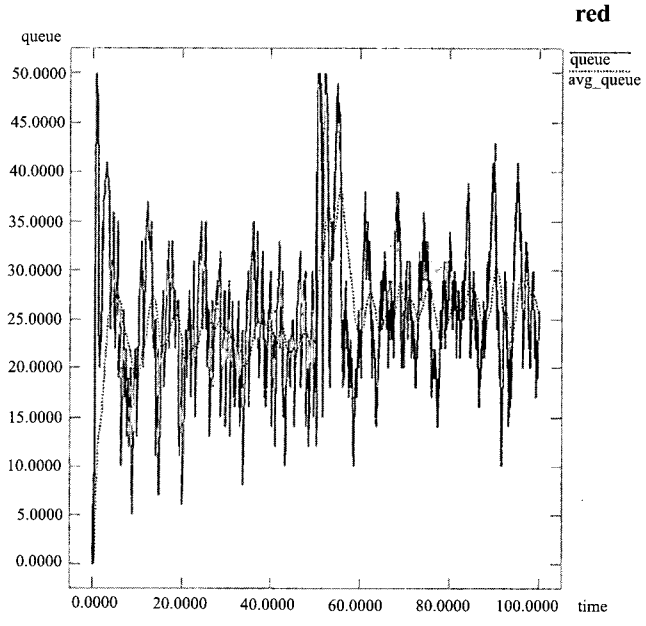


Fig. 14. Fuzzy with an increase in congestion.

as the congestion control window allows them to do so. The receiver immediately sends an acknowledgement message (ACK) packet when it receives a data packet. For both RED and Fuzzy mechanisms, $\max_{th}$, $\min_{th}$, and $w_q$ were set to 80% buffer size, 20% buffer size, and 0.002, respectively.

- **FTP traffic**

In Figs. 7–9, for FTP traffic sources, the queue sizes of RED1, RED0.01, and Fuzzy are plotted versus simulation time. The number of traffic sources was set to 100. All FTP traffic sources start to send packets at the start of simulation. The buffer size is equal to 50 packets. Figs. 7 and 8 show that 1) when $\max_p$ is high, the average queue size is near $\min_{th}$ (RED1), and 2) when $\max_p$ is low, the average queue size is close to $\max_p$ (RED0.01).

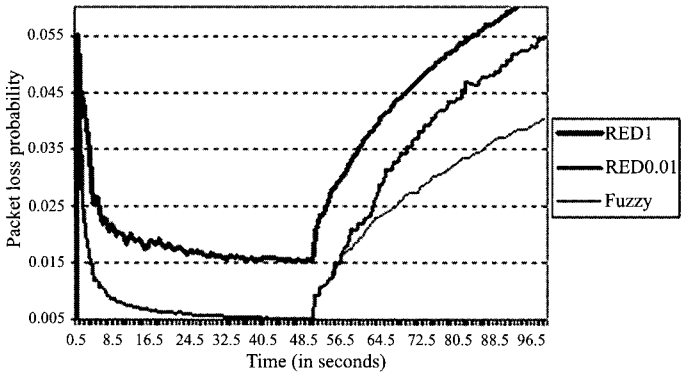


Fig. 15. Packet loss probability of RED1, RED0.01, and Fuzzy versus simulation time with an increase in congestion (buffer size = 50 packets).

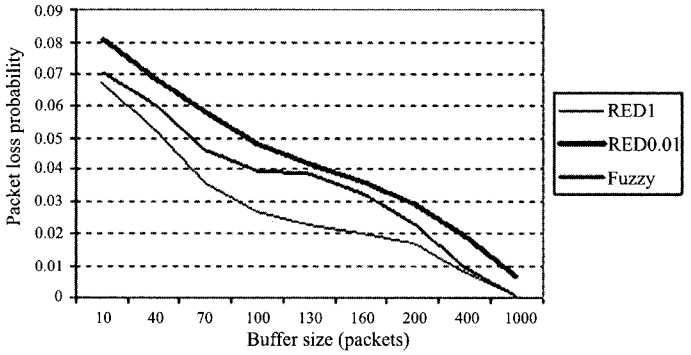


Fig. 16. Packet loss probability of RED1, RED0.01, and Fuzzy versus buffer size with an increase in congestion (buffer size = 50 packets).

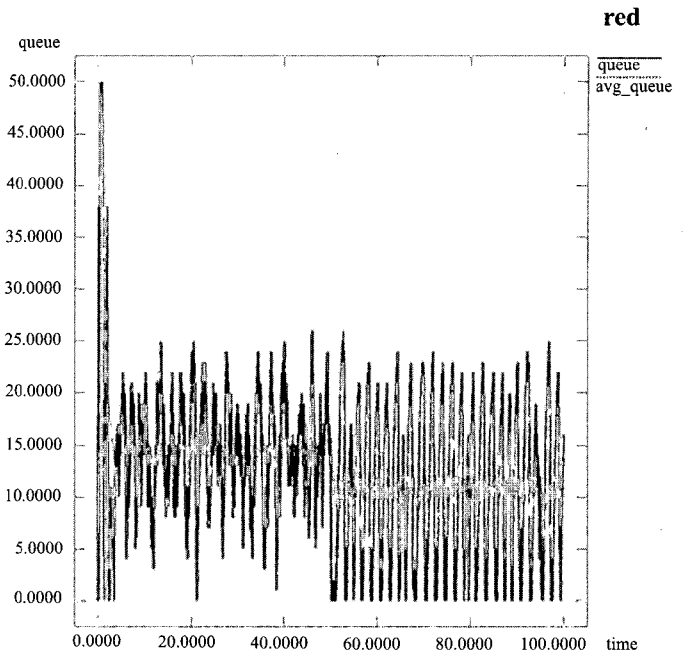


Fig. 17. RED1 with a decrease in congestion.

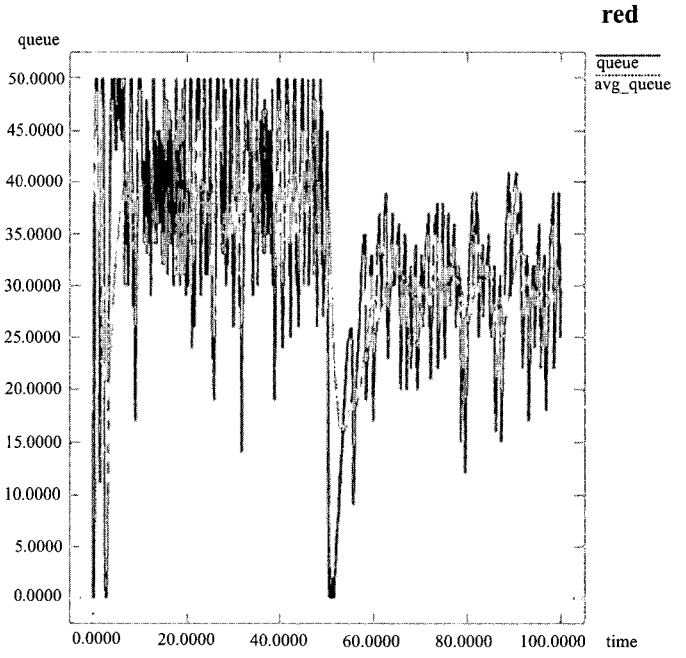


Fig. 18. RED0.01 with a decrease in congestion.

In Fig. 10, the packet loss probability of all mechanisms is plotted versus simulation time. This figure shows the packet loss of our proposed fuzzy controller is 2% less than those of RED1 and RED0.01 mechanisms. In Fig. 11, the packet loss probability of all mechanisms is plotted versus the different number of traffic sources. Based on the results shown in this figure, when the number of traffic sources for all mechanisms is increased, the packet loss probability is increased too. It can be seen that the packet loss probability of the proposed fuzzy controller is 2% less than those of RED1 and RED0.01 mechanisms.

To evaluate the performance of proposed fuzzy controller under different congestion density, more simulations were performed. For the simulation given in Figs. 12–14, four traffic sources started at time 0, and sixteen new traffic sources started at time 50 to send their packets.

The packet loss probability of all mechanisms is shown in Figs. 15 and 16. Fig. 15 shows that for all mechanisms, the packet loss probability is increased at time 50. It can be seen that the packet loss probability of the fuzzy controller is less than those of RED1 and RED0.01 mechanisms. In Fig. 16, for all mechanisms, the packet loss probability is plotted versus different buffer size. The out-performance of the proposed method is easily observed.

In the next simulation, we evaluate the performance of all mechanisms with a decrease in the congestion. For the simulation in Figs. 17–19, twenty traffic sources start at time 0. At time 50, sixteen traffic sources stop sending their packets.

Fig. 20 shows the packet loss probability of RED1, RED0.01, and Fuzzy. As can be seen in this figure, for all mechanisms, the packet loss probability is decreased at time 50. It is clear that the packet loss probability of the proposed fuzzy controller is less than those of RED1 and RED0.01 mechanisms. In Fig. 21, the packet loss probability is plotted at different values of $\min_{th}$ and $\max_{th}$

Figs. 22 and 23, at two values of buffer size, plots the packet loss probability of RED1, RED0.01, and Fuzzy. In this case, fifty FTP sources start to send their packets at the start of simulation.

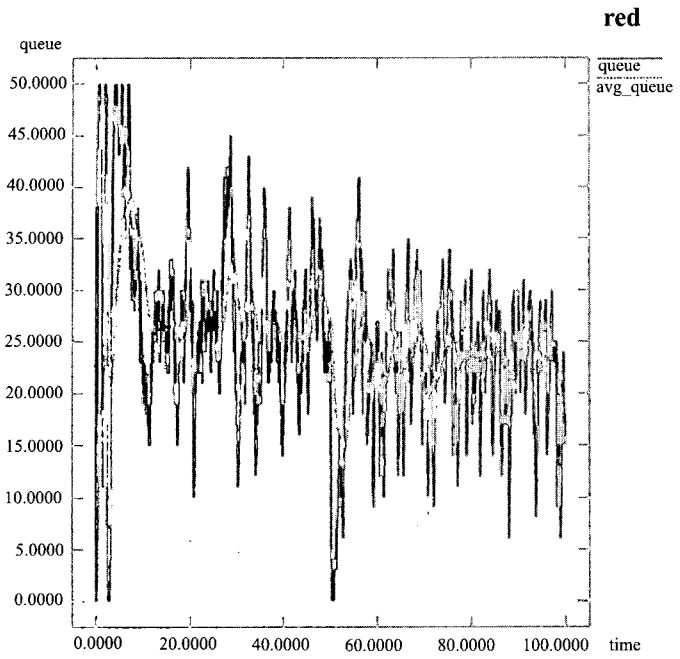


Fig. 19. Fuzzy with a decrease in congestion.

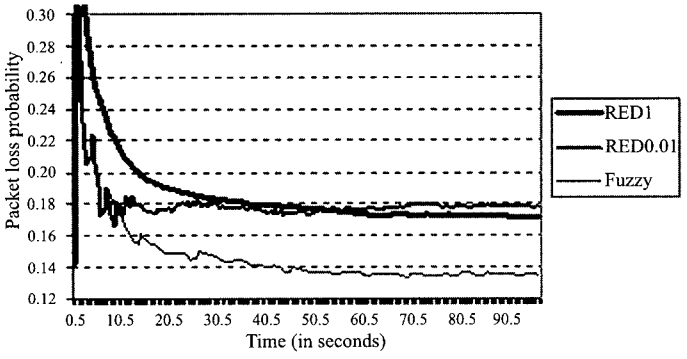


Fig. 20. Packet loss probability of RED1, RED0.01, and Fuzzy versus simulation time (with a decrease in congestion).

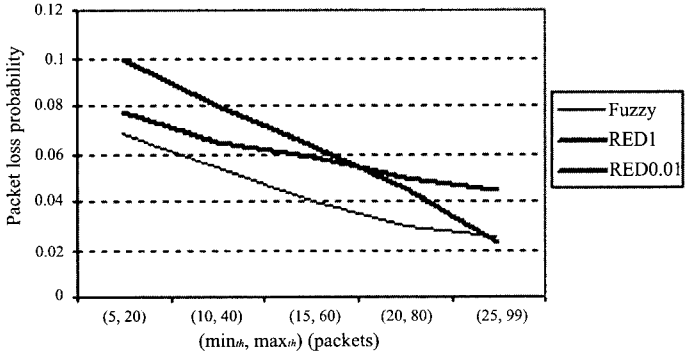


Fig. 21. Packet loss probability at different values of $(\min_{th}, \max_{th})$ (buffer size = 100 packets).

Using the results shown in Figs. 22 and 23, we demonstrate the superiority of the Fuzzy to both RED1and RED0.01 mechanisms in reducing packet losses even when they are all operating with a smaller buffer size. To evaluate the performance of the proposed fuzzy controller at different values of the round trip

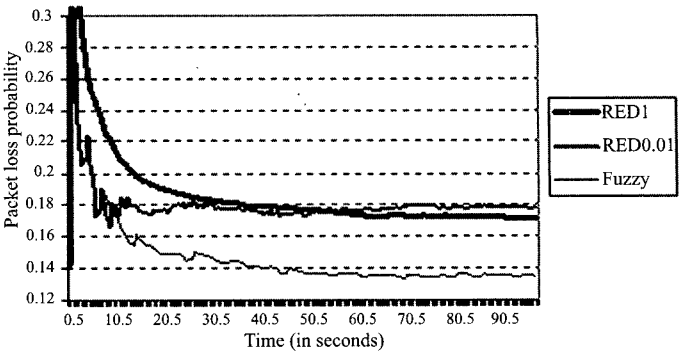


Fig. 22. Packet loss probability of RED1, RED0.01, and Fuzzy at buffer size = 100 packets

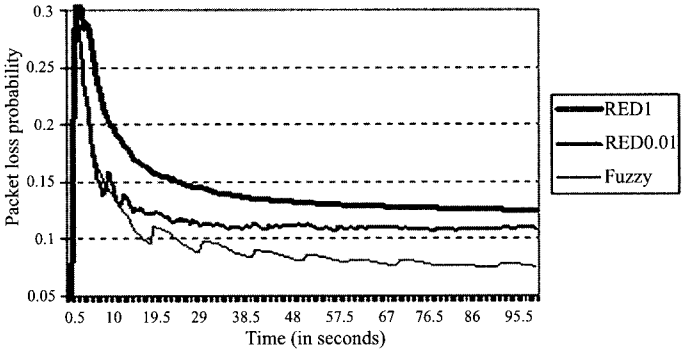


Fig. 23. Packet loss probability of RED1, RED0.01, and Fuzzy at buffer size = 250 packets

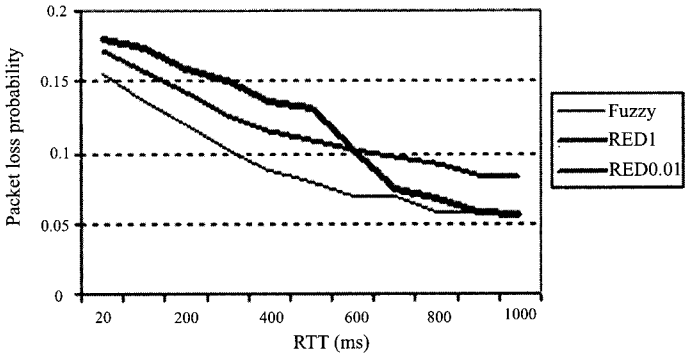


Fig. 24. Packet loss probability of RED 1, RED0.01, and Fuzzy at different value of RTT

time (RTT), a new simulation was performed. In Figs. 24 and 25, for all mechanisms, the packet loss probability and channel utilization are plotted versus different values of the RTT. The buffer size and the number of traffic sources are 100 packets and 50, respectively.

- **Bursty sources**

In this section the performance of the fuzzy controller is evaluated under bursty traffic. For this purpose, an exponentially bursty traffic is simulated in the ns-2. The peak bit rate, mean burst size, mean silence size, and the packet size of bursty traffic sources were set to 1 Mb/s, 0.01 s, 0.1 s, and 1000 bytes, respectively.

In Figs. 26–28, for different values of traffic load, the packet loss probability of all mechanisms is plotted versus simulation

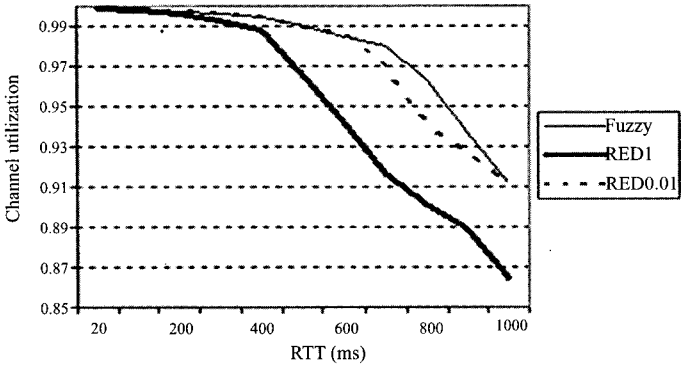


Fig. 25. Channel utilization of RED 1, RED0.01, and Fuzzy at different value of RTT

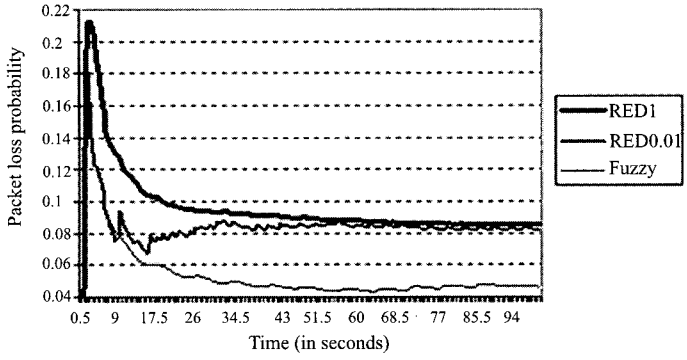


Fig. 26. Packet loss probability of RED1, RED0.01, and Fuzzy at low traffic load (20 traffic sources).

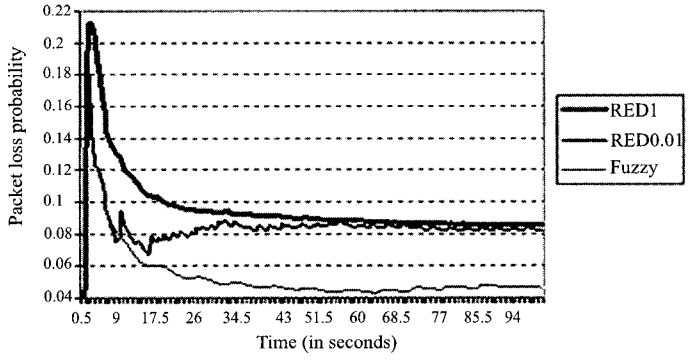


Fig. 27. Packet loss probability of RED1, RED0.01, and Fuzzy at moderate traffic load (50 traffic sources).

time. The buffer size was set to 100 packets. This figure confirms that at different traffic loads, the proposed fuzzy controller has a better packet loss probability than both the RED1 and RED0.01 do.

In Figs. 29–32, for different values of buffer size, the packet loss probability of RED1, RED0.01, and Fuzzy RED is plotted versus simulation time. In this case, at the beginning of the simulation fifty bursty sources start to send their packets. Easily seen in this figure, for all values of the buffer size, the proposed fuzzy controller outperforms the RED1 and RED0.01.

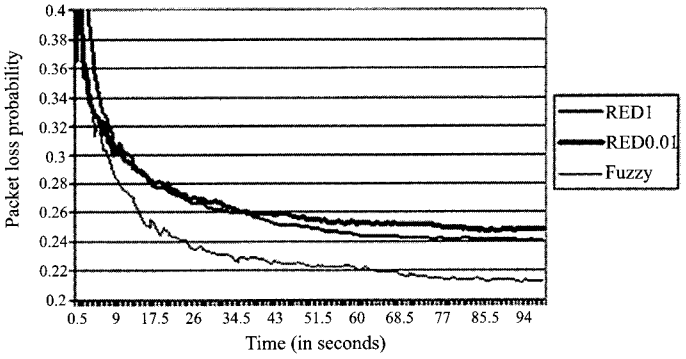


Fig. 28. Packet loss probability of RED1, RED0.01, and Fuzzy at high traffic load (100 traffic sources).

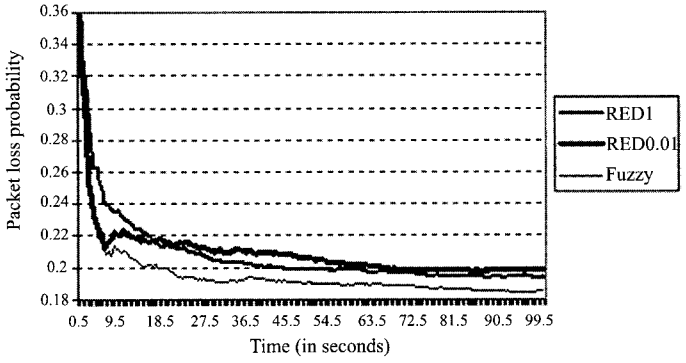


Fig. 29. Packet loss probability of RED1, RED0.01, and Fuzzy at buffer size = 20 packets.

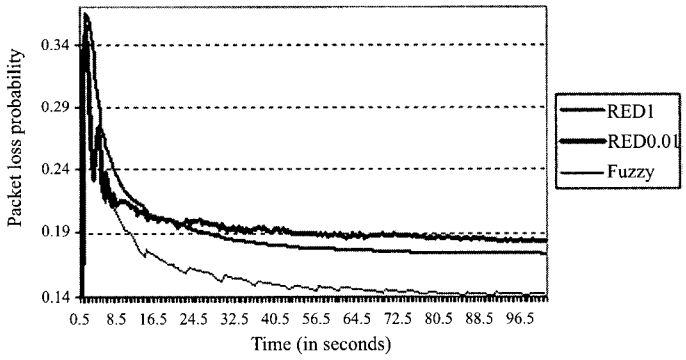


Fig. 30. Packet loss probability of RED1, RED0.01, and Fuzzy at buffer size = 100 packets.

## IV. RELATED WORKS

In [3] and [4], Feng proposed the original ARED mechanism. This mechanism retains RED's basic structure and merely adjusts the parameter $\max_p$ to keep the average queue size between $\min_{th}$ and $\max_{th}$. In [5], a new implementation of the original ARED mechanism was proposed by Floyd that could fix some of its shortcomings. As described in [5], the main objective of the ARED is to keep the average queue size within a target range half way between $\max_{th}$ and $\min_{th}$. To achieve this goal, $\max_p$ is updated slowly in such a way that to stay within the range [0.01, 0.5]. The ARED uses the additive-increase multiplicative decrease (AIMD) policy. In the ARED mechanism, at each periodic time intervals (typically 0.5 seconds), the average

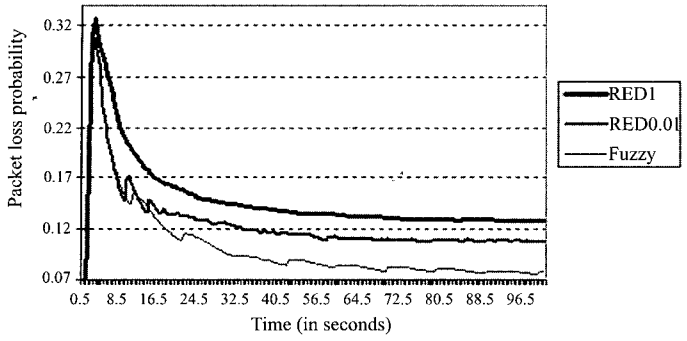


Fig. 31. Packet loss probability of RED1, RED0.01, and Fuzzy at buffer size = 250 packets

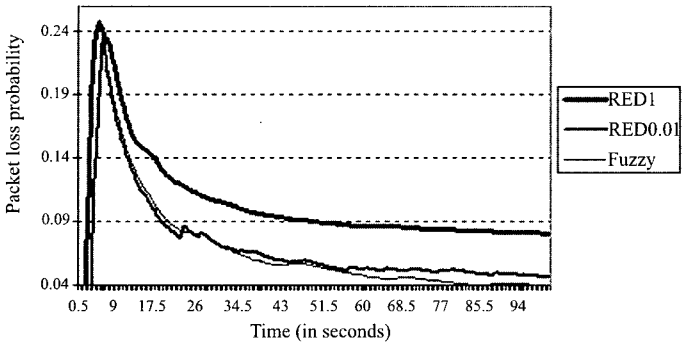


Fig. 32. Packet loss probability of RED1, RED0.01, and Fuzzy at buffer size = 500 packets.

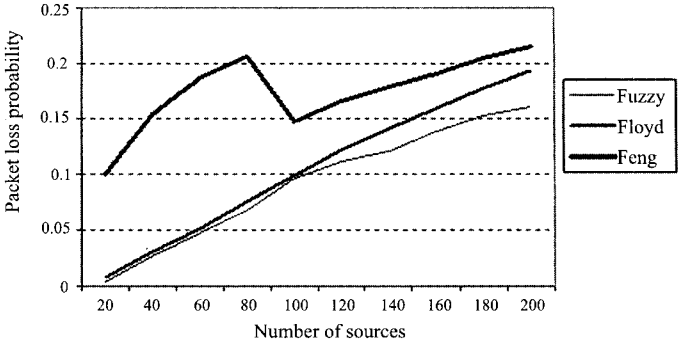


Fig. 33. Packet loss probability of Feng's method, Floyd's method, and Fuzzy at different number of traffic sources.

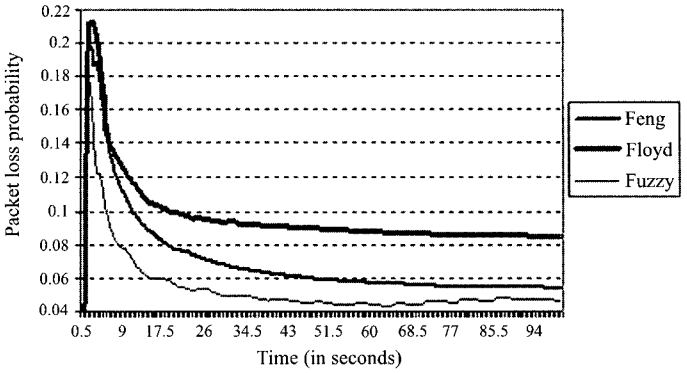


Fig. 34. Packet loss probability of Feng's method, Floyed's method, and Fuzzy at buffer size = 100 packets.

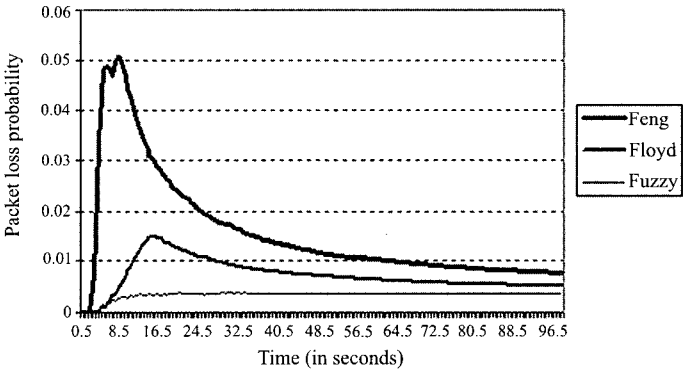


Fig. 35. Packet loss probability of Feng's method, Floyed's method, and Fuzzy at buffer size = 500 packets.

queue size is compared with the target value. When the average queue size is greater than the target and also $\max_p$ is less than 0.5, then $\max_p$ is increased. When the average queue size is less than the target and $\max_p$ is greater than 0.01, then $\max_p$ is decreased. The robustness of the ARED comes from its slow and infrequent adjustments of $\max_p$. When congestion density changes sharply, it could take some time for the ARED mechanism to adapt itself to this new value. In the ARED mechanism, as the value of $\max_p$ stays within the range [0.01, 0.5], its performance will not be degraded during the transition period.

Based on previous studies, the dynamic tuning of $\max_p$ parameter of RED algorithm gives us better performance. In [4], some simulation results illustrat RED's sensitivity to parameters, and show that adaptive RED does indeed address this problem. Based on results shown in [4], the RED's average queue size and its performance are dependent on the RED parameters $\max_p$. Furthermore, it is shown that by adapting $\max_p$ to keep the target queue size within a target range between $\min_{th}$ and $\max_{th}$ it is possible to achieve the good performance.

By combining both flow based and queue based congestion indicators with the capabilities of fuzzy logic controllers, we believe that the performance of the proposed fuzzy controller is better than those of Feng's and Floyd's methods. To prove this, some new trials in the ns-2 simulator are performed. In Fig. 33, for bursty traffic, the packet loss probability of all mechanisms is plotted versus number of traffic sources. The buffer size was set to 500 packets. Based on the results given in the figure, it is clear that the performance of the proposed fuzzy controller is better than those of Feng's and Floyd's mechanisms, especially when the congestion is heavy. For example, when the number of traffic sources is 180, the packet loss probability of Feng's method, Floyed's method and Fuzzy is equal to 0.20, 0.18, and 0.15, respectively. In Figs. 34–36, for different values of the buffer size, the packet loss probability of all mechanisms is plotted versus simulation time. In this case, twenty bursty sources start to send their packets when the simulation begins.

As can be seen in Figs. 34–36, for all value of buffer size, the proposed fuzzy controller shows better performance than Feng's and Floyd's mechanisms.

In Figs. 37 and 38, for all mechanisms, the packet loss probability and channel utilization is plotted versus different values of RTT. In this case, 50 bursty sources start at time 0. The buffer

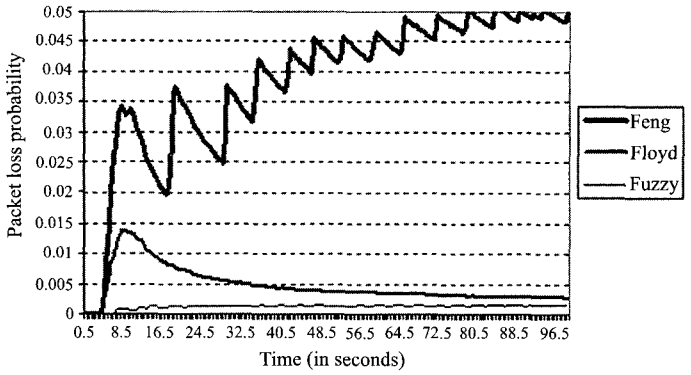


Fig. 36. Packet loss probability of Feng's method, Floyed's method, and Fuzzy at buffer size = 1000 packets.

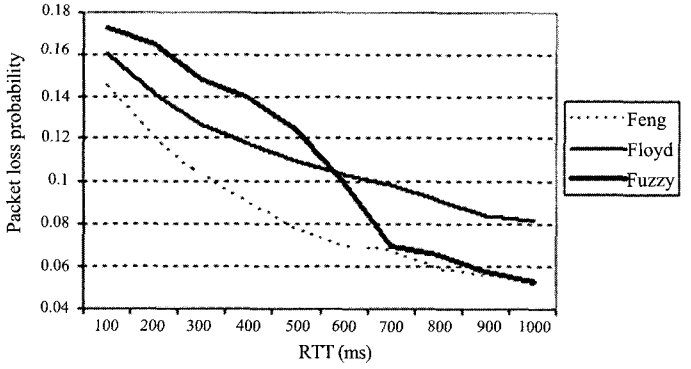


Fig. 37. Packet loss probability of all mechanisms at different value of RTT.

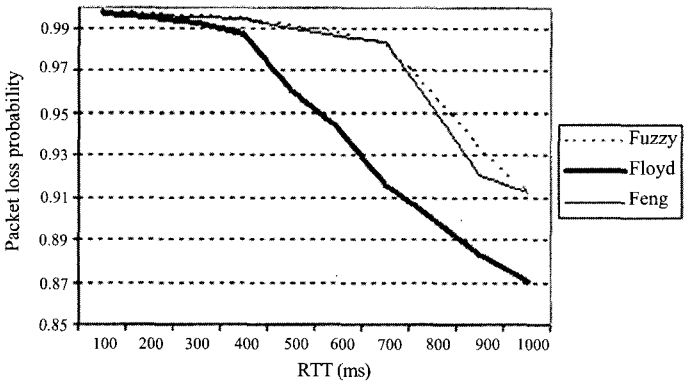


Fig. 38. Channel utilization of all mechanisms at different value of RTT.

size is equal to 100 packets.

To support the claimed results, the performance of the proposed fuzzy controller is evaluated with two bursty traffic sources (data and voice). The values of (peak bit rate, mean burst size, and mean silence size) for data and voice sources are equal to (10 Mb/s, 0.0143 s, and 0.12 s) and (64 kb/s, 0.350 s, and 0.650 s), respectively. Furthermore, to ensure fair bandwidth share in the entire network and not merely one link, the new network topology shown in Fig. 39 was simulated.

In Figs. 40 and 41, for all mechanisms, the packet loss probability and channel utilization are plotted versus number of voice traffic sources. The number of data sources and the buffer size were set to 30 and 100 packets, respectively. Based on the

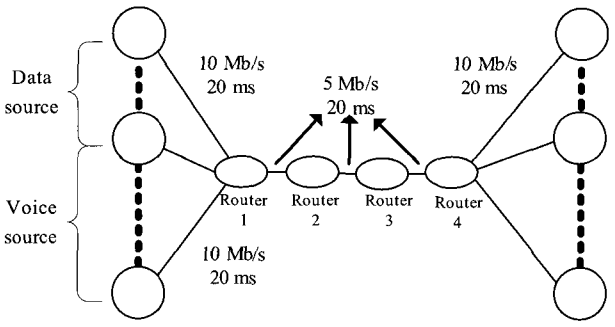


Fig. 39. The second simulation topology.

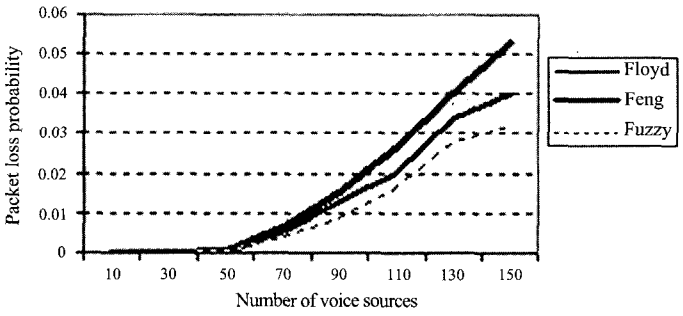


Fig. 40. Packet loss probability of all mechanisms at different number of voice traffic sources.

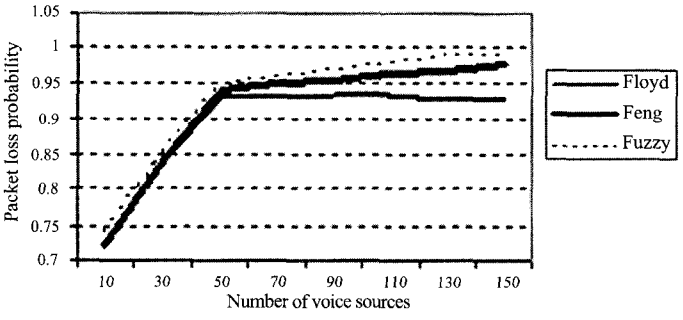


Fig. 41. Channel utilization of all mechanisms at different number of voice traffic sources.

results given in this figure, it is clear that the performance of the proposed fuzzy controller is better than those of Feng's and Floyd's mechanisms.

In Figs. 42 and 43, the packet loss probability and channel utilization are plotted versus number of data traffic sources. The number of voice sources was set to 100.

In Figs. 44 and 45, for all mechanisms, the packet loss probability and channel utilization are plotted versus buffer size. In this case, fifty data traffic sources and one hundred voice traffic sources are connected to the router 1. It can be seen that for all buffer sizes, the proposed fuzzy traffic controller has a better performance in comparison with Feng's and Floyd's mechanisms.

## V. THE PROPOSED FUZZY CONTROLLER FOR DIFFSERV NETWORKS

Differentiated services (DiffServ) [32]–[34], which was proposed by IETF, is an IP QoS architecture based on packet mark-

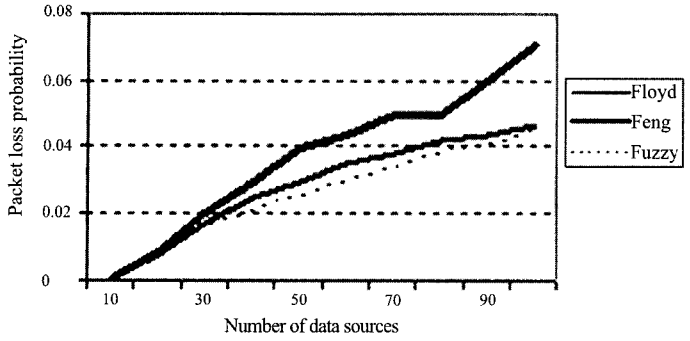


Fig. 42. Packet loss probability of all mechanisms at different number of data traffic sources.

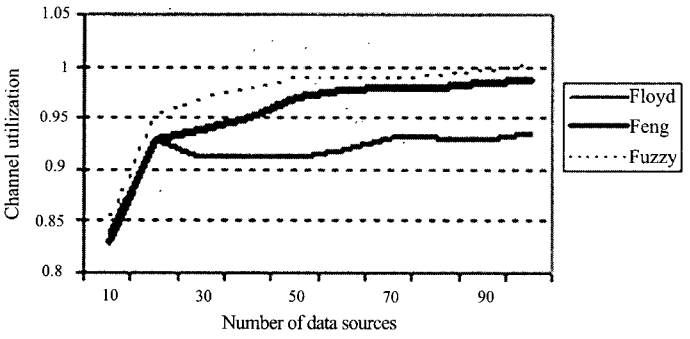


Fig. 43. Channel utilization of all mechanisms at different number of data traffic sources.

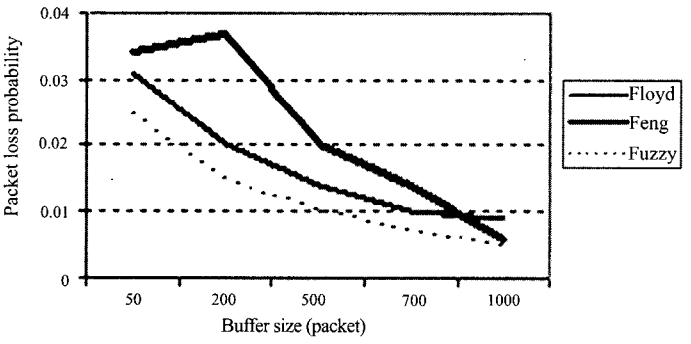


Fig. 44. Packet loss probability of all mechanisms at different buffer size.

ing that allows packets to be prioritized according to user requirements. When congestion occurs in the network, more low priority packets are discarded than high priority packets. Today's best effort model of the Internet can not support the requested QoS requirements of new applications. To solve this problem, the DiffServ model uses several service classes for new applications. The DiffServ architecture provides QoS by dividing traffic into different categories, marking each packet with a code point that indicates its category, and scheduling packets accordingly.

Packets in a single class of traffic are enqueued into one corresponding physical RED queue, which contains three virtual queues (one for each drop precedence). The simulation model, which consists of three virtual queues, is shown in Fig. 46. Different RED parameters can be configured for virtual queues,

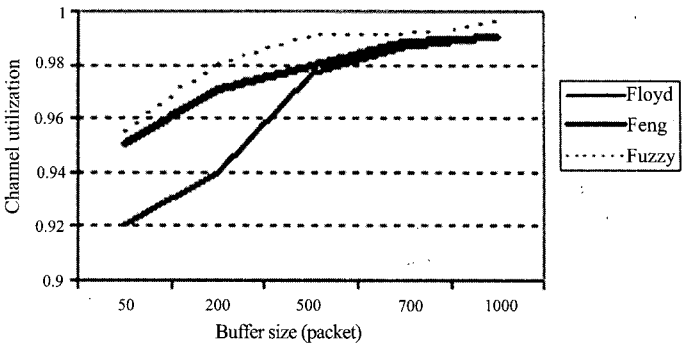


Fig. 45. Channel utilization of all mechanisms at different buffer size.

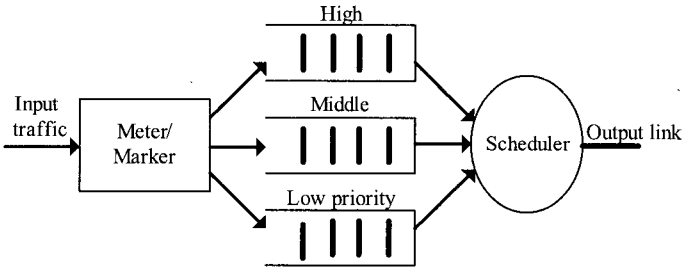


Fig. 46. The simulation model of a DiffServ router.

causing packets from one virtual queue to be dropped more frequently than packets from another. A packet with a lower dropping precedence is given better treatment in times of congestion because it is assigned a code point that corresponds to a virtual queue with relatively lenient RED parameters.

The meter/marker shown in Fig. 46, is responsible for monitoring the arrival times of packets and determining the level of conformance to a pre-established traffic profile.

DiffServ routers use traffic profile which specifies the temporal properties of a traffic stream. The traffic profile includes some traffic parameters such as the mean packet rate, the maximum packet rate, and the maximum burst size. It contains rules for determining whether a particular packet is in-profile or out-of-profile. The concept of in- and out-of-profile can be extended to multiple levels of conformance. In-profile packets may be allowed to enter the network. Out-of-profile packets may be shaped, discarded, marked with a new codepoint (re-marked), or forwarded unchanged.

A DiffServ router may contain the following elements: Meter, marker, shaper, and dropper. A meter is used to measure the traffic stream against a traffic profile. The meter may take a specified action after detecting any violation. It first measures the rate at which packets making up a stream of traffic pass it, then compares the rate to some set of thresholds, and finally produces certain number of potential results. A given packet is said to be "conformant" to a level of the meter if, at the time that the packet is being examined, the stream appears to be within the rate limit for the profile associated with that level. The token bucket (TB) [35], the single rate three colors meter (srTCM) [36], the time sliding window (TSW) approach [37], and the adaptive packet marking [38] are some popular metering/marking mechanisms. The proposed fuzzy model was

Table 1. Simulation parameters.

| | Parameter | TB | TSW | SrTCM |
|---|---|---|---|---|
| High priority queue | $\max_{th}$ | 0.8*BS | 0.8*BS | 0.8*BS |
| | $\min_{th}$ | 0.3*BS | 0.3*BS | 0.3*BS |
| | $\max_p$ | 0.1 | 0.05 | 0.05 |
| Middle priority queue | $\max_{th}$ | - | 0.6*BS | 0.6*BS |
| | $\min_{th}$ | - | 0.2*BS | 0.2*BS |
| | $\max_p$ | - | 0.1 | 0.1 |
| Low priority queue | $\max_{th}$ | 0.5*BS | 0.5*BS | 0.5*BS |
| | $\min_{th}$ | 0.1*BS | 0.1*BS | 0.1*BS |
| | $\max_p$ | 0.2 | 0.2 | 0.2 |
| Committed information rate | CIR | 1000000 | 100000 | 1000000 |
| Peak information rate | PIR | - | 500000 | - |
| Committed burst size | CBS | 3000 | - | 2000 |
| Excess burst size | EBS | - | - | 3000 |

Table 2. Performance of RED and Fuzzy: TB, FTP traffic (high priority queue).

| Buffer size (packets) | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 20 | 16394 | 18186 | 0 | 0 | 0 | 0 |
| 70 | 17642 | 19214 | 0 | 0 | 0 | 0 |
| 150 | 18384 | 20159 | 0 | 0 | 0 | 0 |

Table 3. Performance of RED and Fuzzy: TB, FTP traffic (low priority queue).

| Buffer size (packets) | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 20 | 27865 | 36434 | 759 | 212 | 0.03 | 0.006 |
| 70 | 38184 | 40164 | 148 | 46 | 0.004 | 0.001 |
| 150 | 39592 | 39712 | 72 | 15 | 0.002 | 0.0004 |

Table 4. Performance of RED and Fuzzy: TB, FTP traffic (total).

| Buffer size (packets) | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 20 | 44259 | 54620 | 759 | 212 | 0.02 | 0.004 |
| 70 | 55826 | 59378 | 148 | 46 | 0.003 | 0.0008 |
| 150 | 57976 | 59871 | 72 | 15 | 0.001 | 0.0002 |

extended for DiffServ routers. The performance of fuzzy model is evaluated with 3 different metering/marking mechanisms including: TB, TSW, and srTCM. In Table 1, the values of simulation parameters are given. In this table, BS refers to buffer size. To quantitatively measure the performance of these mechanisms, the following performance ratio for cases in which the number of passed packets are nonzero, is defined:

$$Performence\ Ratio\ (PR) = \frac{Number\ of\ lost\ packets}{Number\ of\ passed\ packets}.$$

Table 5. Performance of RED and Fuzzy: TB, bursty traffic (high priority queue).

| Buffer size (packets) | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 20 | 44700 | 45691 | 624 | 76 | 0.01 | 0.002 |
| 70 | 35856 | 40284 | 1131 | 98 | 0.03 | 0.002 |
| 150 | 36976 | 36416 | 1037 | 1106 | 0.03 | 0.03 |

Table 6. Performance of RED and Fuzzy: TB, bursty traffic (low priority queue).

| Buffer size (packets) | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 20 | 10338 | 12191 | 5145 | 3680 | 0.5 | 0.3 |
| 70 | 23199 | 19673 | 2277 | 2512 | 0.1 | 0.13 |
| 150 | 22250 | 23215 | 1450 | 1182 | 0.06 | 0.05 |

Table 7. Performance of RED and Fuzzy: TB, bursty traffic (total).

| Buffer size (packets) | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 20 | 55038 | 57882 | 5769 | 3756 | 0.1 | 0.06 |
| 70 | 59055 | 59957 | 3408 | 2610 | 0.06 | 0.04 |
| 150 | 59226 | 59631 | 2487 | 2288 | 0.04 | 0.04 |

Table 8. Performance of RED and Fuzzy for TSW (high priority queue).

| Buffer size (packets) | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 10 | 4759 | 4806 | 0 | 0 | 0 | 0 |
| 70 | 4890 | 4850 | 0 | 0 | 0 | 0 |
| 150 | 4866 | 4852 | 0 | 0 | 0 | 0 |

- **The TB scheme**

The goal of the TB scheme is to allow a source to send a stream of "in" packets at a rate equal to the target rate with a predetermined burst size. The marker uses the following rule: A packet is marked as "in-profile", if there is a token available, otherwise it is sent as "out-of-profile". In Tables 2–4, for FTP traffic and under different values of the buffer size, the performance of Fuzzy and RED mechanisms are given. The number of traffic sources are equal to 20. As shown in these tables, for both low and high priority queue, the proposed fuzzy model has a better channel utilization and packet loss probability as well. For example, when the buffer size is equal to 20 packets, the total performance ratio of Fuzzy and RED mechanisms are 0.004 and 0.02, respectively. In this case, for the RED mechanism, the channel utilization is 30% less than that of the fuzzy RED mechanism.

Tables 5–7 summarize the results of the simulation for bursty traffic. In this case, 20 bursty sources are connected to the router. It is clear that for any buffer size, the channel utilization of the fuzzy model is higher than that of the RED mechanism, and the

Table 9. Performance of RED and Fuzzy for TSW (middle priority queue).

| Buffer size (packets) | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 10 | 18954 | 18908 | 21 | 16 | 0.001 | 0.0008 |
| 70 | 18870 | 18872 | 0 | 0 | 0 | 0 |
| 150 | 18759 | 18885 | 0 | 0 | 0 | 0 |

Table 10. Performance of RED and Fuzzy for TSW (low priority queue).

| Buffer size (packets) | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 10 | 26119 | 28020 | 803 | 358 | 0.03 | 0.01 |
| 70 | 33628 | 36066 | 78 | 9 | 0.002 | 0.0002 |
| 150 | 35029 | 36132 | 61 | 13 | 0.002 | 0.0003 |

Table 11. Performance of RED and Fuzzy for TSW (total).

| Buffer size (packets) | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 10 | 49832 | 51734 | 824 | 374 | 0.02 | 0.007 |
| 70 | 57388 | 59788 | 78 | 9 | 0.001 | 0.0001 |
| 150 | 58654 | 59869 | 61 | 13 | 0.001 | 0.0001 |

loss probability of the Fuzzy is lower than that of the RED. For example for a high priority queue, for a buffer size of 70 packets, the performance ratio of Fuzzy and RED are equal to 0.002 and 0.03, respectively.

**•The TSW scheme**

The goal of the TSW algorithm is to automatically allocate the channel bandwidth in a certain region during periods of congestion. The bandwidth denoted by a single parameter is specified by the service level agreement. In Tables 8–11, for FTP traffic and under different buffer size, the performance of proposed Fuzzy model and RED is given. In this case, twenty FTP traffic sources are connected to the router. Based on results shown in these tables, it is clear that for any buffer size and for any priority, the performance of the Fuzzy is better than that of the two RED mechanisms. For example, at a low buffer size (10 packets), the total performance ratio of Fuzzy and RED is 0.007 and 0.02, respectively. In this case, the channel utilization of fuzzy model is 1.04 times that of the RED mechanism.

Now we consider the case in which the number of traffic sources changes. Tables 12–15 list the results of the simulation for this case. The buffer size was set to 100 packets. The results confirm that at any number of sources and for all level of priorities, the Fuzzy outperforms remarkably the RED mechanism.

**• The srTCM scheme**

The srTCM, marks the input IP packets green, yellow, or red. Marking is based on a committed information rate (CIR) and two associated burst sizes, or a committed burst size (CBS) and an excess burst size (EBS). The behavior of the srTCM meter is specified in terms of two token buckets, C and E, which both share the common rate CIR. The maximum size of the token buckets C and E are CBS and EBS, respectively.

In Tables 16–19, for FTP traffic sources and at different number of sources, the performances of both mechanisms are shown. The buffer size was set to 50 packets. For example, when the number of traffic sources is equal to 4, the total performance ratio of Fuzzy and RED are equal to 0.0002 and 0.07, respectively. Furthermore, the Fuzzy model transmits 6562 more packets than the RED mechanism.

Table 12. Performance of RED and Fuzzy for TSW (high priority queue).

| Number of sources | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 4 | 4773 | 4814 | 0 | 0 | 0 | 0 |
| 8 | 9098 | 9326 | 59 | 85 | 0.006 | 0.009 |
| 12 | 13498 | 13550 | 100 | 262 | 0.007 | 0.02 |
| 16 | 17669 | 17919 | 401 | 505 | 0.02 | 0.02 |
| 20 | 22037 | 22160 | 759 | 773 | 0.03 | 0.03 |

Table 13. Performance of RED and Fuzzy for TSW (middle priority queue).

| Number of sources | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 4 | 18857 | 18883 | 0 | 0 | 0 | 0 |
| 8 | 32747 | 35052 | 627 | 288 | 0.02 | 0.008 |
| 12 | 38345 | 39350 | 1894 | 790 | 0.05 | 0.02 |
| 16 | 37425 | 40081 | 2096 | 1108 | 0.05 | 0.03 |
| 20 | 36752 | 37330 | 1820 | 1488 | 0.05 | 0.04 |

Table 14. Performance of RED and Fuzzy for TSW (low priority queue).

| Number of sources | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 4 | 32620 | 35999 | 102 | 40 | 0.003 | 0.001 |
| 8 | 10695 | 13167 | 342 | 146 | 0.03 | 0.01 |
| 12 | 2048 | 4909 | 276 | 68 | 0.13 | 0.01 |
| 16 | 623 | 772 | 56 | 107 | 0.09 | 0.14 |
| 20 | 18 | 150 | 14 | 6 | 0.77 | 0.04 |

Table 15. Performance of RED and Fuzzy for TSW (total).

| Number of sources | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 4 | 56250 | 59696 | 102 | 40 | 0.002 | 0.0007 |
| 8 | 52540 | 57545 | 1028 | 519 | 0.02 | 0.009 |
| 12 | 53891 | 57809 | 2270 | 1120 | 0.04 | 0.02 |
| 16 | 55717 | 58772 | 2553 | 1720 | 0.04 | 0.03 |
| 20 | 58797 | 59640 | 2593 | 2267 | 0.04 | 0.04 |

## VI. CONCLUSIONS

Random early detection (RED) is the most important AQM mechanism proposed in the literature to solve problems caused by the drop tail (DT) queue management mechanism. Based on previous research activities, the traditional RED algorithm contains severe problems. The performance of the RED is too sensitive to the traffic load and its parameter settings. In this paper, a fuzzy logic control approach for active queue management is presented. In the proposed model, a fuzzy logic controller is used to dynamically tune the dropping probability of

Table 16. Performance of RED and Fuzzy for srTCM (high priority queue).

| Number of sources | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 4 | 22103 | 15449 | 31 | 0 | 0.001 | 0 |
| 8 | 19063 | 18692 | 1 | 4 | 0.00005 | 0.0002 |
| 12 | 22103 | 22533 | 31 | 23 | 0.001 | 0.001 |
| 16 | 24302 | 26487 | 128 | 32 | 0.005 | 0.001 |
| 20 | 28671 | 30985 | 29 | 41 | 0.001 | 0.001 |

Table 17. Performance of RED and Fuzzy for srTCM (middle priority queue).

| Number of sources | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 4 | 13041 | 6520 | 697 | 0 | 0.05 | 0 |
| 8 | 10810 | 11103 | 49 | 12 | 0.004 | 0.001 |
| 12 | 13041 | 15338 | 697 | 24 | 0.05 | 0.001 |
| 16 | 14480 | 18907 | 1275 | 50 | 0.09 | 0.003 |
| 20 | 22504 | 21421 | 193 | 127 | 0.008 | 0.006 |

Table 18. Performance of RED and Fuzzy for srTCM (low priority queue).

| Number of sources | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 4 | 18041 | 37778 | 3058 | 12 | 0.17 | 0.0003 |
| 8 | 23792 | 28009 | 2752 | 876 | 0.11 | 0.03 |
| 12 | 18041 | 21107 | 3058 | 1367 | 0.17 | 0.06 |
| 16 | 15038 | 14234 | 2716 | 2023 | 0.18 | 0.14 |
| 20 | 6013 | 7364 | 4847 | 3047 | 0.8 | 0.41 |

Table 19. Performance of RED and Fuzzy for srTCM (total).

| Number of sources | Passed packets | | Lost packets | | Performance ratio | |
|---|---|---|---|---|---|---|
| | RED | Fuzzy | RED | Fuzzy | RED | Fuzzy |
| 4 | 53185 | 59747 | 3786 | 12 | 0.07 | 0.0002 |
| 8 | 53665 | 57804 | 2802 | 892 | 0.05 | 0.01 |
| 12 | 53185 | 58978 | 3786 | 1414 | 0.07 | 0.02 |
| 16 | 53820 | 59628 | 4119 | 2105 | 0.08 | 0.03 |
| 20 | 57188 | 59770 | 5069 | 3215 | 0.09 | 0.05 |

the RED mechanism. The main objective of the model is to tune the $\max_p$ parameter of the RED mechanism so that its average queue size remains nearly constant. The performance of the proposed fuzzy controller was measured for two types of traffic sources, including FTP and bursty traffic. It was observed that the proposed fuzzy controller can protect the QoS of TCP connections while simultaneously it perfectly utilized the network resources. The performance of fuzzy controller was compared with those of the Feng's and the Floyed's adaptive RED mechanisms too. In this case, it was also observed that the proposed model is clearly superior to their models. Finally the proposed fuzzy control model was applied to the DiffServ networks. The simulation results helped us to judge the merit of the proposed fuzzy mechanism.

## REFERENCES


[1] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[2] The Network Simulator—ns-2 homepage, available at http://www.isi.edu/nsnam/ns/.

[3] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Techniques for eliminating packet loss in congested TCP/IP network," *U. Michigan CSE-TR-349-97*, Nov. 1997.

[4] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A self configuring RED gareway," in *Proc. IEEE INFOCOM'99*, Mar. 1999.

[5] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED," *Technical Report*, 2001.

[6] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM'98*, Aug. 1998, pp. 314–329.

[7] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons not to deploy RED," in *Proc. IWQoS'99*, June 1999, pp. 260–262.

[8] W. Feng, D. Kandlur, D. Saha, and K. Shin, "BLUE: A new class of active queue management algorithms," *U. Michigan CSE-TR-387-99*, Apr. 1999.

[9] T. Ott, T. Lakshman, and L. Wong, "SRED: Stabilized RED," in *Proc. IEEE INFOCOM'99*, 1999.

[10] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active queue management," *IEEE Network*, May/June 2001.

[11] B. Wydrowski and M. Zukerman, "GREEN: An active queue management algorithm for a self managed Internet," in *Proc. IEEE ICC 2002*, New York, vol. 4, 2002, pp. 2368–2372.

[12] R. Pan, V. Prabhakar, and K. Psounis, "Choke: A stateless active queue management scheme for approximating fair bandwidth allocation," in *Proc. IEEE INFOCOM 2000*, Tel Aviv, Israel, vol. 2, Mar. 2000, pp. 942–951.

[13] S. Ghosh, Q. Razouqi, H. J. Schumacher, and A. Celmins, "A survey of recent advances in fuzzy logic in telecommunications networks and new challenges," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 3, pp. 443–447, 1998.

[14] B. Bensaou, S. T. C. Lam, H.-W. Chu, and D. H. K. Tsang, "Estimation of the cell loss ratio in ATM networks with a fuzzy system and application to measurement-based call admission control," *IEEE/ACM Trans. Networking*, vol. 5, no. 4, pp. 572–584, 1997.

[15] V. Catania, G. Ficili, S. Palazzo, and D. Panno, "A comparative analysis of fuzzy versus conventional policing mechanism for ATM networks," *IEEE/ACM Trans. Networking*, vol. 4, pp.449–459, 1996.

[16] T. D. Ndousse, "Fuzzy neural control of voice cells in ATM networks," *IEEE J. Select. Areas Commun.*, vol. 12, no. 9, pp.1488–1494, 1994.

[17] C. Douligeris and G. Develekos, "A fuzzy logic approach to congestion control in ATM networks," in *Proc. IEEE ICC'95*, Seattle, WA, vol. 3, June 1995, pp. 1969–1973.

[18] A. Pitsillides and Y. A. Sekercioglu, "Fuzzy logic control of cell loss in asynchronous transfer mode (ATM)," in *Proc. Australian Telecommun. Network App. Conf.*, Clayton, Australia, Dec. 1994, pp. 249–254.

[19] R.-G. Cheng and C.-J. Chang, "Design of a fuzzy traffic controller for ATM networks," *IEEE Trans. Networking*, vol. 4, pp. 460–469, 1996.

[20] M. H. Yaghmaee, M. Safavi, and M. B. Menhaj, "A novel fuzzy traffic controller for ATM networks," in *Proc. IEEE ISPACS'98*, Melbourn, Australia, 1998, pp. 199–203.

[21] M. H. Yaghmaee, M. Safavi, and M. B. Menhaj, "A high performance fuzzy traffic controller for ATM networks," in *Proc. IEEE ATM Workshop'99*, Kochi, Japan, 1999, pp. 379–384.

[22] M. H. Yaghmaee, M. Safavi, and M. B. Menhaj, "A fuzzy connection admission controller for ATM networks," in *Proc. IEEE ISPACS'99*, Pucket, Thailand, 1999.

[23] M. H. Yaghmaee, M. Safavi, and M. B. Menhaj, "An efficient fuzzy based traffic policer for ATM networks," *IEICE Trans. Commun.*, vol. E83-B, no.1, pp. 10–19, 2000.

[24] M. H. Yaghmaee, M. Safavi, and M. B. Menhaj, "An intelligent usage parameter controller based on dynamic rate leaky bucket for ATM networksm," *Elsevier Science J. Computer Networks*, vol. 32, pp. 17–34, 2000.

[25] M. H. Yaghmaee, M. Safavi, and M. B. Menhaj, "A novel FLC based approach for ATM traffic control," *Elsevier Science J. Computer Networks*, vol. 36, no. 5/6, pp. 643–658, 2001.

[26] A. R. Bonde and S. Ghosh, "A comparative study of fuzzy versus fixed threshold for robust queue management in cell-switching networks," *IEEE/ACM Trans. Networking*, vol. 2, no. 4, pp. 337–344, 1994.

[27] M. P. Fernandez, A. de C. P. Pedroza, and J. F. de Rezende, "QoS provisioning across a DiffServ domain using policy-based management," in *Proc. GLOBECOM 2001*, San Antonio, USA, Nov. 2001.

[28] C. Chrysostomou, A. Pitsillides, G. Hadjipollas, A. Sekercioglu, and M. Polycarpou, "Fuzzy logic congestion control in TCP/IP best-effort networks," in *Proc. ICTTA 2004*, Damascus, Syria, Apr. 2004, pp. 19–23.

[29] M. P. Fernandez, A. de C. P. Pedroza, and J. F. de Rezende, "Optimizing fuzzy controllers with genetic algorithms for QoS improvement," in *Proc. ITS 2002*, Natal, Brazil, 2002.
[30] C. Chrysostomou1, A. Pitsillides, L. Rossides, M. Polycarpou, and A. Sekercioglu, "Congestion control in differentiated services networks using fuzzy-RED," *IFAC J. Control Eng. Practice*, vol. 11, no. 10, pp. 1153–1170, 2003.
[31] L. X. Wang, *A Course in Fuzzy Systems and Control*, NJ: Prentice-Hall, 1997.
[32] S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *IETF RFC 2475*, 1998.
[33] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," *IETF RFC 2638*, July 1999.
[34] T. Li and Y. Rekhter, "A provider architecture for differentiated services and traffic engineering (PASTE)," *IETF RFC 2430*, Oct. 1998.
[35] J. S. Turner, "New direction in communication," in *Proc. Int. Zurich Seminar Digital Commun.*, Zurich, 1986.
[36] J. Heinane and R. Guerin, "A single rate three color marker," *IETF RFC 2697*, Sept. 1999.
[37] W. Fang, N. Seddigh, and B. Nandy, "A time sliding window three color marker (TSWTCM)," *IETF RFC 2859*, June 2000.
[38] W. Feng, D. D. Kandlur, D. Saha, and K. Shin, "Adaptive packet marking for providing differentiated services in the Internet," in *Proc. Int. Conf. Network Protocols*, Austin, TX, 1998, pp. 108–117.

**Mohammad Hossien Yaghmaee** was born on July 1971 in Mashad, Iran. He received his B.S. degree in Communication Engineering from Sharif University of Technology, Tehran, Iran in 1993, and M.S. degree in communication engineering from Tehran Polytechnic (Amirkabir) University of Technology in 1995. He received his Ph.D. degree in communication engineering from Tehran Polytechnic (Amirkabir) University of Technology in 2000. He has been a computer network engineer with several networking projects in Iran Telecommunication Research Center (ITRC) since 1992. He is author of one book: Computer Networks and Internet, in persian, (Ferdowsi University of Mashad Publishing, Mashad, 2003). His research interests are in traffic and congestion control, high speed networks including ATM and MPLS, quality of services (QoS), and fuzzy logic control.