Application of Fuzzy Logic Controllers in Traffic Management of Communication Networks

Mohammad Hossien Yaghmaee[1], Mohammad Bagher Menhaj[2]

ABSTRACT

During past few years, Fuzzy Logic Controllers have found many applications in different area of engineering. In this paper, we present the applications of these controllers in the traffic management of different communication network technology including Asynchronous Transfer Network and Internet Protocol. Different fuzzy controllers will be proposed for different traffic management mechanisms including buffer management, Active Queue Management, traffic policing, traffic metering, scheduling, Usage Parameter Control and Call Admission Control. Based on simulation results, it can be seen that all proposed fuzzy controllers demonstrate better performance in comparison with traditional mechanisms.

KEYWORDS

Communication Networks, Traffic Management, Fuzzy Logic Control, Quality of Service

I. INTRODUCTION

During the early 1990s, the Asynchronous Transfer Mode (ATM), as specified by CCITT, was considered as a new high speed technology for a replacement of the Ethernet standard at higher speeds. The basic goal of ATM was to define a networking technology around the basic idea of fast packet switching. Since new services were expected to operate at multi-megabit rates, a fast packet switching technology was desired. The ATM forum defined different traffic parameters for describing traffic that is injected into the ATM network at the User Network Interface (UNI) [1]. These traffic parameters are: Peak Cell Rate (PCR), Cell Delay Variation Tolerance (CDVT), Sustainable Cell Rate (SCR), Maximum Burst Size (MBS), and Minimum Cell Rate (MCR).

The ATM Forum defined different service classes to be supported by ATM. The Quality of Service (QoS) parameters are defined for all conforming cells of a connection, where conformance is defined with respect to a Generic Cell Rate Algorithm (GCRA) described in the ATM Forum User-Network Interface Specification 3.1. The input to this algorithm is the traffic parameters. The proposed service categories by the ATM forum are then described as follows:

- CBR (Constant Bit Rate): the CBR service class is intended for real-time applications, i.e., those requiring tightly constr ined delay and delay variation, as would be appropriate for voice and video applications.
- VBR-rt (Variable Bit Rate – Real Time): the real-time VBR service class is intended for real-time applications, i.e., those requiring minimal loss and tightly constrained delay and delay variation, as would be appropriate for voice and video applications.
- VBR-nrt (Variable Bit Rate – Non-Real Time): the non-real time VBR service class is intended for non-real time applications which have "bursty" traffic characteristics and which can be characterized in terms of a GCRA.
- UBR (Unspecified Bit Rate:) the UBR service class is intended for delay-tolerant or nonreal-time applications, i.e., those which do not require tightly constrained delay and delay variation, such as traditional computer communications applications.
- ABR (Available Bit Rate): many applications have the ability to reduce their information transfer rate if the network requires them to do so.

[1] M. H. Yaghmaee is with Department of Engineering, Ferdowsi University of Mashad, Mashad, Iran (e-mail: hyaghmae@ferdowsi.um.ac.ir).
[2] M. B. Menhaj is with Department of Electrical Engineering, Amirkabir University of Technology, Tehran, Iran (e-mail: menhaj@aut.ac.ir).

In the 1990s, while ATM technology was being developed to integrate voice, data, and video, pure data services embraced the TCP/IP protocol, or the IP technology. What made the IP technology attractive is its universal adoption due mainly to the popularity of the global Internet and the unprecedented growth rates the Internet has reached. Initially, IP was not designed for the integration of voice, data, and video to the end user. IP defines the basic data unit of data transfer and it also performs the routing function. Therefore, IP is also referred to as the layer 3 protocol in the Internet suite as it corresponds to the layer 3 (network layer) of the OSI model. However, currently a packet is transported in the Internet without any guarantees to its delay or loss. Due to this "best effort" forwarding paradigm, the Internet cannot provide integrated services over this infrastructure. Several QoS architectures are proposed by the IETF for IP networks to enable the support of integrated services over IP networks. The integrated services (intserv) architecture [2] defines a set of extensions to the traditional best effort model of the Internet so as to provide end-to-end QoS commitments to certain applications with quantitative performance requirements. This service is intended for adaptive real time applications that can tolerate a certain amount of loss and delay provided it is kept to a reasonable level. The integrated services architecture assumes some explicit setup mechanism such as RSVP (Resource Reservation Protocol) [3]-[4]. This setup or signaling mechanism will be used to convey QoS requirements to IP routers so that they can provide requested services to flows that request them. Upon receiving per-flow resource requirements through RSVP, the routers apply intserv admission control to signaled requests. The routers also enable traffic control mechanisms to ensure that each admitted flow receives the requested service independent of other flows. One of the reasons that have impeded the deployment of integrated services with RSVP is the use of per-flow state and per flow processing which typically exceeds the flow-handling capability of today's core routers.

In contrast with the per-flow nature of integrated services, differentiated services (diffserv) networks[5]-[6] classify packets into one of a small number of aggregated flows or "classes" based on the Diffserv Code point (DSCP) written in the differentiated services field of the packet's IP header. This is known as Behavior Aggregate (BA) classification. At each diffserv router in a Diffserv Domain (DS domain), packets receive a Per Hop Behavior (PHB), which is invoked by the DSCP. Differentiated services are extended across a DS domain boundary by establishing a Service Level Agreement (SLA) between an upstream network and a downstream DS domain. Traffic classification and conditioning functions (metering, shaping, policing, remarking) are performed at this boundary to ensure that traffic entering the DS domain conforms to the rules specified in the Traffic Conditioning Agreement (TCA) which is derived from the SLA. A PHB then refers to the packet scheduling, queuing, policing, or shaping behavior of a node on any given packet belonging to a BA, as configured by a service level agreement or a policy decision. The following standard PHBs are defined:

- Default PHB: provides a best-effort service in a diffserv compliant node
- Assured Forwarding (AF) PHB [7]: the AF PHB group defines four AF classes: AF1, AF2, AF3, and AF4. Each class is assigned a specific amount of buffer space and interface bandwidth, according to the SLA with the service provider or a policy decision. Within each AF class, three drop precedence values are assigned.
- Expedited Forwarding (EF) PHB [8]: EF PHB provides a guaranteed bandwidth service with low loss, delay and delay jitter. EF PHB can be implemented with priority queuing and rate limiting on the behavior aggregate.

During the past few years, fuzzy logic has been found many applications in different area of telecommunication networks including: Call Admission Control (CAC) and Usage Parameter Control (UPC) in ATM networks, adaptive buffer management [9], policy-based management [10], congestion control in TCP/IP best-effort networks and differentiated services.

In this paper, we present some applications of fuzzy logic in different area of traffic management of ATM and IP networks. For this purpose, we propose the following fuzzy based controllers:

- Fuzzy Dynamic Rate Leaky Bucket (FDRLB): This mechanism is presented for traffic policing in ATM networks [11].
- Fuzzy Connection Admission Control (FCAC): The FCAC decides to accept or reject the requested connection. The new connection is only accepted if the network has enough resources to provide the QoS requirements of the new connection without affecting the QoS of the already established connections [12].
- Fuzzy Usage Parameter Control (FUPC): FUPC is responsible to monitor the traffic stream for detecting any possible violation from traffic contract [13].
- Fuzzy Meter: The FM measures the rate at which IP packets are entered to the DS domain [14].
- Fuzzy Random Early Detection (FRED): FRED is fuzzy implementation of traditional RED mechanism which is used to detect and react to the congestion before its consequences such as packet loss or queuing delays [15].
- Fuzzy BLUE (FBLUE): The main idea of FBLUE is to use different parameters such as packet loss and link utilization

to detect congestion and adjust the rate of packet dropping/marking [16].

- Fuzzy TCP: The Fuzzy TCP is an implementation of traditional TCP Vegas mechanism.

Based on simulation results it can be seen that all proposed fuzzy mechanisms have a better performance than the traditional non-fuzzy mechanisms.

The remainder of this paper is organized as follows. Section 2 describes the proposed fuzzy controllers for traffic management in ATM networks. In this section, we introduce FDRLB, FCAC and FUPC. In section 3, we present different fuzzy controllers for differentiated services. Finally, section 4 concludes the paper.

II. APPLICATION OF FUZZY CONTROLLERS IN THE ATM NETWORKS

In ATM networks, many traffic sources with different traffic characteristics and different QoS parameters, share the network resources through statistical multiplexing. The QoS requirements are specified as the network performance parameters including cell loss ratio, cell transfer delay and cell delay variations. Traffic characteristics of an ATM connection are described by traffic parameters such as: peak cell rate, mean cell rate, cell delay variation tolerances, burstiness and maximum burst size. A set of these traffic parameters construct the source traffic descriptor which is used during the connection establishment to capture the intrinsic traffic characteristics of the requested connection. At the network side, Connection Admission Control (CAC) procedures, decide to accept or reject the requested connection. The new connection is only accepted if the network has enough resources to provide the QoS requirements of the new connection without affecting the QoS of the already established connections. After connection establishment phase, the traffic parameters and QoS values agreed by both traffic source and network form a traffic contract. During data transfer phase, to avoid any congestion situation, it is necessary for network to monitor the traffic stream for detecting any possible violation from traffic contract. This function is done by Usage Parameters Control (UPC). The UPC function is performed at the User-Network Interface (UNI) to protect network resources from any unintentional misbehavior of traffic source.

If traffic source generates excessive traffic cells, then the violating traffic cells can be dropped (hard control) or tagged (soft control). The most important UPC mechanisms which have been proposed in the literature are: Leaky Bucket (LB) [12]-[24], Jumping Window (JW) [19], Triggered Jumping Window (TJW) [19], Moving Window (MW) [19] and Exponentially Weighted Moving Average (EWMA) [19],[25].

A. Fuzzy Dynamic Rate Leaky Bucket (FDRLB)

The Leaky Bucket (LB) algorithm is the most popular traffic politer in the ATM network. The LB consists of a counter incremented by arrival of each cell and decrement periodically by a leaky rate, a_r. When the counter reaches a given threshold, N, then the arriving cells are discarded. By using the traffic parameters the counter size, N, and leaky rate, a, are determined at the call setup phase.

In the proposed fuzzy controller we have combined the fuzzy logic with Dynamic Rate Leaky Bucket (DRLB) algorithm [26]. The DRLB is consisting of a data buffer and a credit pool. The DRLB operates like the LB algorithm with exception that the credit generation in the DRLB is changed dynamically according to the state of traffic source. In the active period of the source, the credit generation rate is at the maximum rate R_0 and when source is in the off period, the rate is the minimum value R_1. By using the traffic parameters the credit pool size M and buffer size K are determined. If we assume that the source generates an average L bits in the ON period and S bit in the off period with the mean rate r then the steady state probability γ that the source is in on period is given by: $\gamma = \frac{r}{S}$. The DRLB loads ρ_0, ρ_1, ρ are defined as: $\rho_0 = \frac{r}{R_0}$, $\rho_1 = \frac{r}{R_1}$, $\rho = \frac{\mathrm{r}}{\mathrm{R}}$ where R is the mean credit generation. By using this parameters the cell loss probability, P_{loss}, of the DRLB is calculated as below [26] :

$$P_{loss} = \left(\frac{1}{\gamma} - \frac{1}{\rho_1}\right) \frac{\left[\rho_0 \gamma \ (1-\rho_1) + \gamma \ (1-\gamma)(\rho_1 - \rho_0)\right] e^{\lambda(M+K)}}{\rho_1(1-\gamma) - \rho_0(\rho_1 - \gamma) e^{\lambda(M+K)}} \tag{1}$$

where λ is calculated as below:

$$\lambda = -\frac{\rho_1(1-\rho_1) + \gamma \ (\rho_0 - \rho_1)}{L(1-\gamma \ )(\rho_1 - \gamma)} \tag{2}$$

The proposed Fuzzy Dynamic Rate Leaky Bucket (FDRLB) has 3 crisp inputs which are: the current state of bucket (x), the normalized estimated mean bit rate of the source ($y = \frac{\hat{m}}{m_0}$) and another crisp input z which shows that observed leaky rate is greater or less than the nominal leaky rate, a_r. To measure the estimated source mean bit rate ($\hat{m}$), we count the input cells in a specific time window T. The normalized estimated mean bit rate ($y = \frac{\hat{m}}{m_0}$) and the state of bucket (x) is converted into two fuzzy subsets. If a_k represent the leaky rate at the *k'th* time window then the average leaky rate $\hat{a}$ is calculated as:

$$\hat{a} = \frac{\sum_{i=1}^{k} a_i}{k} \tag{3}$$

The average leaky rate $\hat{a}$ have to be less than or equal to nominal leaky rate, a, which is determined at the call setup phase. We use a crisp input z which demonstrates that average leaky rate $\hat{a}$ is more or less than the nominal leaky rate, a, and is given by: $z = \hat{a} - a$. The change value in leaky rate, a, after k time window, Δa_k, is calculated by using the hight defuzzification method. The new leaky rate a_k after k time window is given by:

$$a_k = a + \Delta \mathrm{a}_\mathrm{k} \tag{4}$$

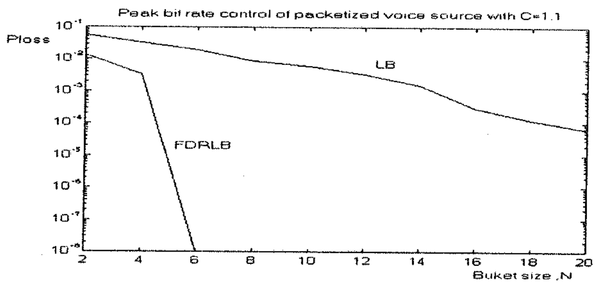
Simulation Results

In this section by using the simulation results which are obtained from packetized voice traffic sources, we compare the performance of our proposed algorithm, FDRLB, with the conventional LB. For the peak bit rate control the nominal leaky rate, a, must be near the peak bit rate, b_0, and bucket size, N, must be very low [27]. On the other hand, for mean bit rate control the leaky rate must be near the source mean bit rate, m_0, and bucket size must be very high. Figure 1(a,b) for peak bit rate control and at two different overload factor C=1.1 and C=1.2,) show the cell loss probability of FDRLB and LB versus bucket size, N. In each case, we see that the cell loss probability of the proposed FDRLB is less than conventional LB.

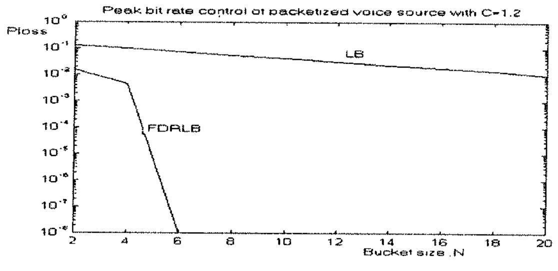
Figure 2, for mean bit rate control at overload factor C=1, shows the cell loss probability of FDRLB and LB versus bucket size, N. In this case we see that for well behavior source the cell loss probability of FDRLB is less than conventional LB.

B. Fuzzy Connection Admission Control (FCAC)

In the ATM network by using the signaling procedures at the call setup phase, a traffic contract has agreed between traffic source and ATM network. The Connection Admission Control (CAC) procedures, decide to accept or reject the requested connection.



(a)



(b)

Figure 1: Cell loss probability of FDRLB and LB for peak bit rate control in two cases, (a) C=1.1 ; (b) C=1.2

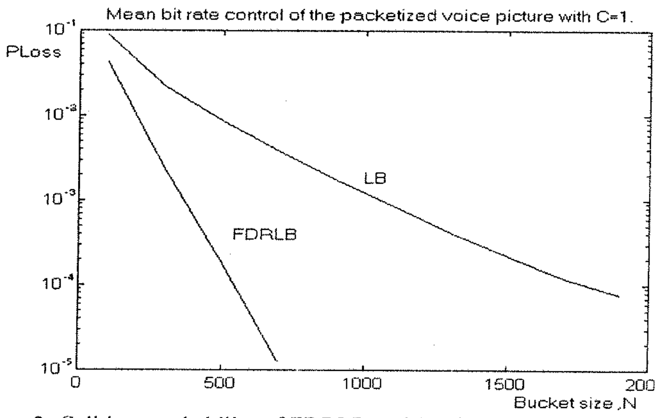
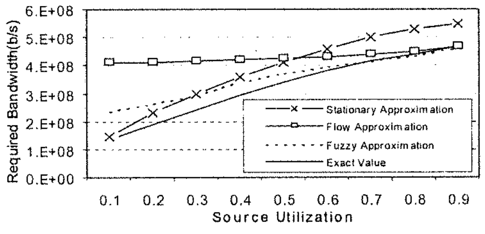


Figure 2: Cell loss probability of FDRLB and LB for mean bit rate control with C=1.

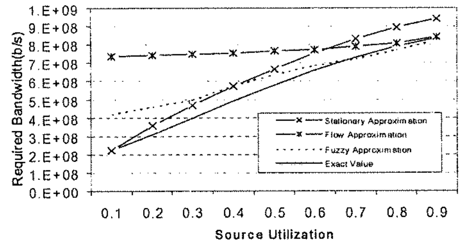
The new connection is only accepted if the network has enough resources to provide the QoS requirements of the new connection without affecting the QoS of the already established connections. The proposed FCAC, which is a modified version of equivalent capacity method [22], uses a three input-single output fuzzy controller. The inputs of FCAC are: peak bit rate, source utilization and the burst period of traffic source. The output of fuzzy controller is used to estimate the required bandwidth of all existing connection.

Simulation Results

To evaluate the efficiency of FCAC, many computer simulation trials were performed. We compare the accuracy of the fuzzy approximation of the required capacity with that of fluid flow approximation and stationary approximation. For this purpose, two types of bursty traffic sources, including packetized voice and data sources were simulated. To measure the performance parameters, an ATM multiplexer has been simulated. The multiplexer is modeled as a finite queue served by a single server with First In First Out (FIFO) service discipline. In Figure 3(a,b), for different number of data traffic sources and for different mechanisms, the required bandwidth is plotted versus source utilization. The peak bit rate and mean burst time are equal to 10Mb/s and 0.03 s, respectively.



(a)



(b)

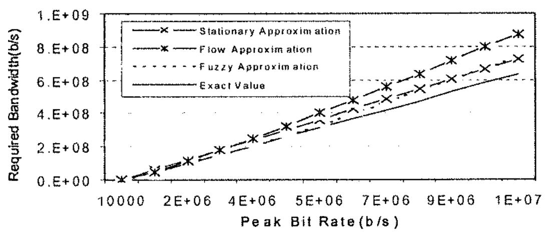
Figure 3: The required capacity versus source utilization, (a) number of traffic sources =50; (b) number of traffic sources =90

In Figure 4(a,b), for different value of source utilization and for different mechanisms, the required capacity is plotted versus peak bit rate. The number of data sources and the mean burst time are set to 90 and 0.03 s, respectively.

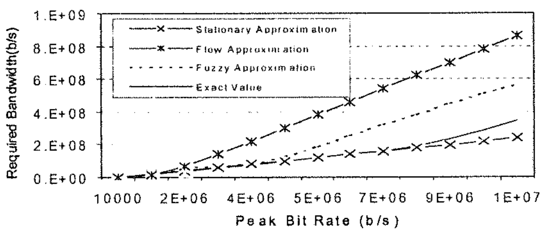
As shown in Figures 3-4, it can be seen that the proposed fuzzy approximation has a better performance in comparison with stationary and flow approximations.

C. Fuzzy Usage Parameter Control (FUPC)

In the ATM networks, when a new connection is accepted by the CAC procedure, the traffic parameters and QoS values agreed by both traffic source and network form a traffic contract. During data transfer phase, to avoid any congestion situation, it is necessary for network to monitor the traffic stream for detecting any possible violation from traffic contract. This function is done by usage parameters control. The UPC function is performed at the user-network interface (UNI) to protect network resources from any unintentional misbehavior of traffic source. If traffic source generates excessive traffic cells, then the violating traffic cells can be dropped (hard control) or tagged (soft control). The proposed FUPC consists of two important parts including a Fuzzy Policer (FP) and a Fuzzy Tagging Rate Controller (FTRC).



(a)



(b)

Figure 4: The required capacity versus peak bit rate, (a) source utilization =0.5 ; (b) source utilization =0.1

The proposed FP, which is based on dynamic rate LB, uses a closed-loop fuzzy control system, to change the leaky rate according to the state of traffic source. By real time estimation of arrival mean cell rate, the leaky rate is changed so that the detected excessive cell is equal to generated excessive. The main objective of FTRC is to decide whether to tag or discard the violating cells according to the state of network congestion.

The FP is based on dynamic rate LB which leaky rate is dynamically tuned according to the state of traffic source. The FP consists of five important parts including: the burst/silence length measurer, generated excessive load estimator, detected excessive load estimator, a fuzzy controller and an LB. The input traffic is monitored by the burst/silence length measurer. This unit is responsible to measure the number of cells in the burst and also the length of silence phase. At the end of each burst/silence period, the measured values of number of cells in the burst and also silence length are sent to the generated excessive load estimator. The detected excessive load estimator, by counting the number of detected violating cells in a fixed time window, estimates the excessive load of traffic source. At the end of each burst silence period, the difference between the outputs of generated excessive load estimator and detected load estimator is sent to the input of the fuzzy controller. The output of fuzzy controller is used to calculate the new value of LB service time.

The second part of IUPC is the FTRC which according to the state of network congestion dynamically tunes the tagging rate. We use the buffer occupancy of multiplexer, as a network congestion indicator. At each time window T1, the buffer occupancy of multiplexer is observed by the FTRC. When the number of cells in the multiplexer buffer is low, the tagging rate becomes high and most of violating cells are tagged and entered to the network. Conversely, when the occupancy of multiplexer is high, then the tagging rate is low and most of violating cells are discarded. Figure 5 shows the location of FCAC and FUPC in an ATM network.

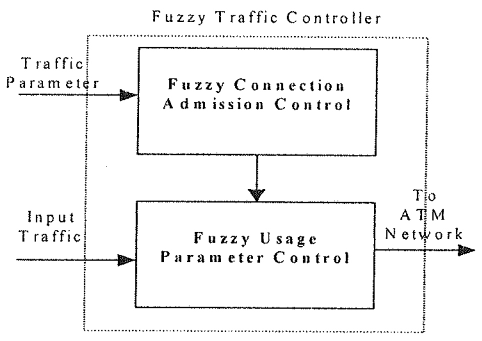


Figure 5: The proposed location of proposed FCAC and FUPC

Simulation Results

To evaluate the performance of FUPC, many simulations with data and voice sources were performed. The performance of FP was compared with that of LB, EWMA, JW and ideal policer. Figure 6 shows the selectivity performances of FP and non-fuzzy policing mechanisms for various normalized mean cell rate. As shown in this figure, the FP has a null false alarm probability and a detection probability very close to ideal policer.

To study the performance of the proposed FTRC, two LB mechanisms with soft and hard control called by Tagging LB and Discarding LB, respectively, are simulated. We also consider combining Tagging LB with Discarding LB. This scheme is called Combined LB. In the Combined LB, when the number of cells in the multiplexer buffer falls below the fixed threshold TH1, then all violating cells are tagged. When the queue length of the multiplexer exceeds the second threshold TH2, then all violating cells are discarded. Figure 7(a) shows the cell loss probability of well-behavior voice connections versus different values of excessive load of violating voice connections. The total load offered by well-behavior connections is equal to 0.59. As shown in this figure ,when the excessive load increases, then for all UPC mechanisms, the cell loss probability of well-behavior connections also increases but for the proposed fuzzy traffic controller the intensity of increasing is lower than those of the non-fuzzy mechanisms.

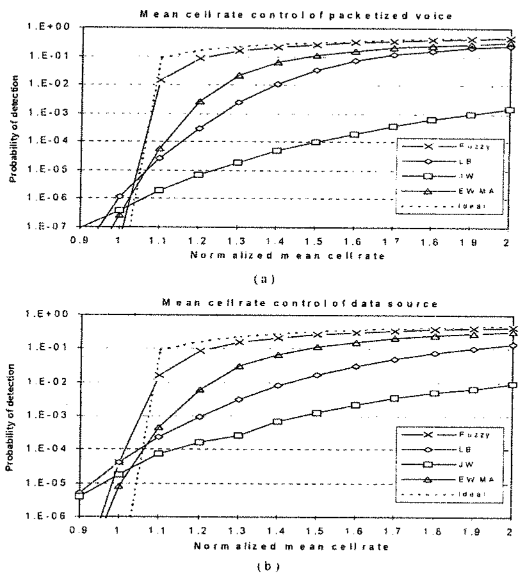


Figure 6: Selectivity performance versus cell rate variation for: (a) voice sources; (b) data sources

As shown in this figure, when the excessive load increases then for fuzzy traffic controller the cell loss probability of well behavior connections is nearly constant while for Tagging LB, Discarding LB and Combined LB, the cell loss probability increases. The channel utilization of the proposed fuzzy traffic controller is shown in Figure 7(b) and compared with Tagging LB, Discarding LB and Combined LB. This figure shows that Tagging LB due to preventing any possible loss at the UNI interface has the better channel utilization than both Discarding LB and Combined LB. But as it is shown in Figure 7(a), the negative aspect of Tagging LB is that it cannot protect the QoS of well-behavior connections. It is observed that the proposed fuzzy traffic controller has channel utilization nearly equal to that of Discarding LB and Combined LB.

In Figure 7(c), the average cell delay of well-behavior voice connections is plotted versus the excessive traffic load of violating voice sources. It can be easily observed that the Discarding LB due to its low channel utilization has the better delay performance than the other mechanisms.

III. APPLICATION OF FUZZY CONTROLLER IN THE DIFFERENTIATED SERVICES

The Internet Engineering Task Force (IETF) has proposed many service models and mechanisms to support the requested QoS in the Internet. The diffserv model is one of the most important models for supporting QoS in the Internet. In the diffserv model, input packets are marked differently to create several packet classes. Each class has different QoS requirements. The diffserv model provides service classification by means of the Differentiated Service (DS) field in the IP header and the Per-Hop Behavior (PHB) which defines the externally observable behavior at the node. The given treatment to the packet within the node is identified by the PHB. The defined PHB are Expedited Forwarding (EF), Assured Forwarding (AF) and Default Behavior (DE) (the existing best effort traffic). The EF PHB provides a low-loss, low-jitter and low-delay handling within the diffserv router. The AF PHB group is used to provide reliable services even during congestion in the network. Currently four AF PHB groups denoted as AF1 to AF4 are defined. The AF PHB uses scheduling mechanisms for resource management.

As described in [6], a disffserv router consists of the following components:

- **Classifier:** packet classifier selects packets in a traffic stream based on the content of some portion of the packet header.
- **Meter:** the meter is responsible to measure the temporal properties of the traffic stream selected by a classifier against a traffic profile specified in a Traffic Conditioning Agreement (TCA). A meter passes state information to other conditioning functions to trigger a particular action for each packet which is either in- or out-of-profile.
- **Marker:** the packet marker sets the DS field of a packet to a particular code point, adding the marked packet to a particular DS behavior aggregate. The marker may be configured to mark all packets which are steered to it to a single code point, or may be configured to mark a packet to one of a set of code points used to select a PHB in a PHB group, according to the state of a meter. When the marker changes the code point in a packet, it is said to have "re-marked" the packet.
- **Shaper:** The shaper delays some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. A shaper usually has a finite-size buffer, and packets may be discarded if there is not sufficient buffer space to hold the delayed packets.
- **Dropper:** The dropper discards some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. This process is know as "policing" the stream. In the next subsection, we present the application of fuzzy system in the different area of diffserv networks.

A. Fuzzy Metering

A meter measures the rate at which packets making up a stream of traffic pass it, compares the rate to some set of thresholds and produces some 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. Some examples of possible meters are: Leaky Bucket (LB), single rate Three Color Meter (srTCM) [28], two rate Three Color Meter (trTCM) [29], time sliding window approach [30] and adaptive packet marking [31].

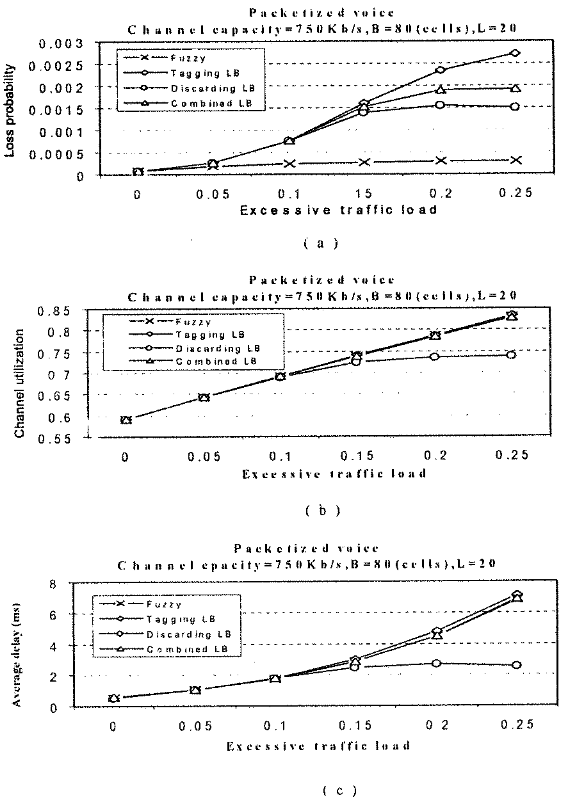


Figure 7: Performance comparisons for voice traffic source 2: (a) cell loss probability of well-behavior connections versus excessive traffic load; (b) channel utilization versus excessive traffic load; (c) average cell delay of well-behavior connections versus excessive traffic load.

The proposed Fuzzy Meter uses a two-input-single-output fuzzy controller. The inputs of the fuzzy controller are: the estimated mean burst size and the average output queue size. Based on linguistic rules stored in the rule base, the fuzzy controller determines the color of the output packet.

Figure 8 shows a block diagram of our proposed fuzzy mechanism. As shown in this figure, our fuzzy controller has 2 crisp inputs which are: the current average buffer size (avg) and the estimated burst size of the traffic source (BS).

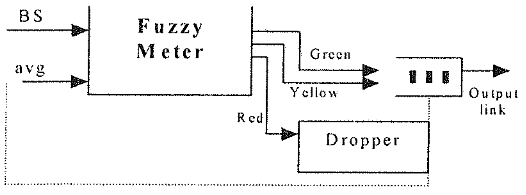


Figure 8: Structure of Fuzzy Meter

Simulation Results

To evaluate the performance of the proposed Fuzzy Meter with that of non-fuzzy mechanism, we consider three conformance levels including: Green, Yellow and Red. The IP packets which are marked as Red, are immediately dropped. Only the Green and Yellow packets are entered to the output buffer. During periods of congestion, in the output buffer, Green packets are dropped with lower probability than Yellow packets. For Green and Yellow packets, we compare the loss probability, the channel utilization and the queuing delay of proposed fuzzy metering/marking mechanism with those of non-fuzzy mechanism. A bursty data traffic source is used for simulation. The packet size is set to 256 bytes. The number of packet per burst has a geometric distribution and the silence phase has an exponential distribution.

In the simulation trials, 250 data traffic sources are connected to a T4 multiplexer, so the traffic load is equal to 0. 9. In Figure 9(a,b), for two traffic classes Green and Yellow and for both non-fuzzy and the proposed fuzzy mechanisms, the packet loss probability is plotted versus the buffer size. It is clear that for Green's packets, in comparison with non-fuzzy mechanism, our proposed fuzzy mechanism has better packet loss probability and also for low priority packets (Yellow's packets), its packet loss probability is close to non-fuzzy mechanism. In Figure10 (a,b,c,d), for two traffic classes Green and Yellow and for both non-fuzzy and the proposed fuzzy mechanisms, the channel utilization and queuing delay are plotted versus the buffer size. This figure shows that the proposed fuzzy mechanism has better performance than non fuzzy mechanisms.

B. Fuzzy Random Early Detection (FRED)

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. By using AQM mechanisms, the senders are informed early about congestion and can react accordingly. Random Early Detection (RED) [32] 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 [33-34], 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 [34]. Different variants of RED such as SRED [35] and Adaptive RED (ARED) [36]-[38] have been proposed which could fix some of its shortcomings. In [39-[40] some AQM mechanisms were proposed which use flow based congestion indicator.

In the proposed Fuzzy RED, we use a fuzzy logic controller to reduce the loss probability of the RED mechanism. The main objective of our proposed model is to tune the loss probability of RED mechanism so that its average queue size remains nearly constant. Our proposed fuzzy controller has two inputs and single output. The inputs linguistic variables are: 1- The error signal e1 which is calculated as the difference between average queue size *(avg)* and a target point, 2- The error signal e2 which is calculated as the difference between the estimated incoming data rate and the target link capacity.

The output of fuzzy controller is used to calculate the new value of *max_p*. The main objective of our proposed fuzzy controller is to control the average queue size near a target point. When the *avg* is less than the target and the incoming data rate is less than the target link capacity, the *max_p* is decreased which decreases the loss rate. On the other hand, when *avg* is greater than the target and also the incoming data rate is greater than the target link capacity, the *max_p* is increased which increases the loss rate. By controlling the *max_p* dynamically, the proposed fuzzy mechanism can achieve a low loss rate.

Simulation Results

To compare the performance of Fuzzy RED with that of traditional RED mechanism, we added our proposed fuzzy controller as a new queue management algorithm to the ns-2

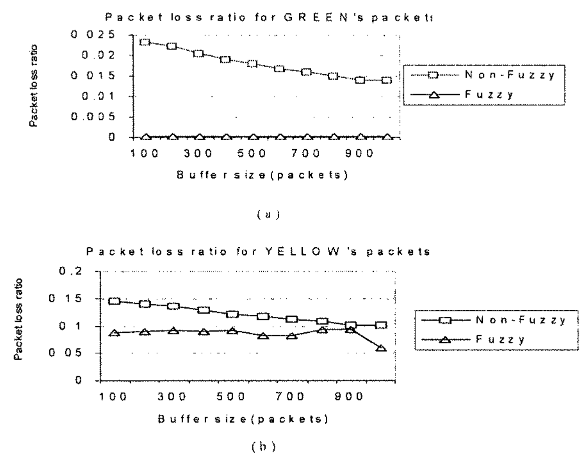


Figure 9: Packet loss versus buffer size for: (a) Green packets; (b) Yellow packets

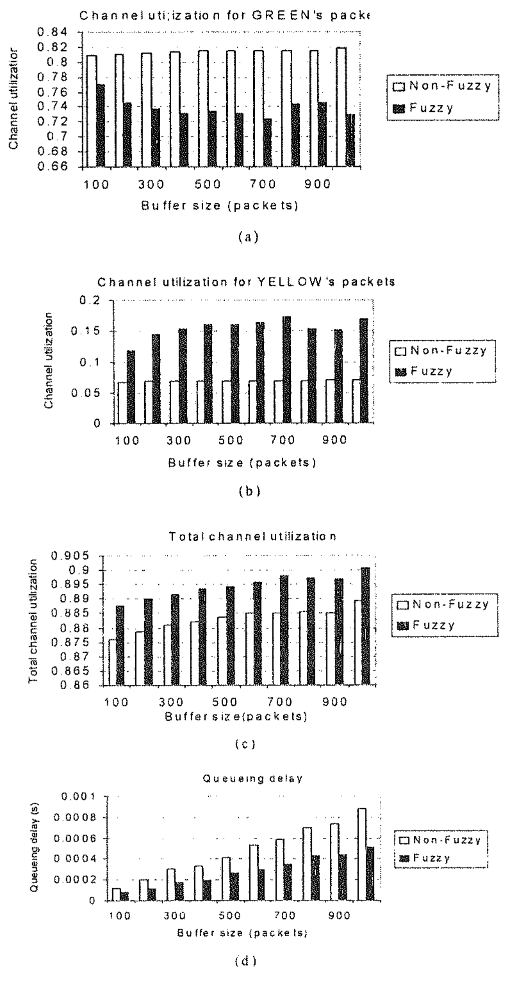


Figure 10: Performance evaluation of fuzzy and non-fuzzy mechanisms: (a) channel utilization of Green packets; (b) channel utilization of Yellow packets; (c) total channel utilization; (d) queuing delay.

[41] simulator. To show the sensitivity of RED algorithm to *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 Figure 11. As shown in this figure, some TCP traffic sources are directly connected to a network router. To evaluate the performance of fuzzy controller, we consider both FTP and bursty traffics. For FTP traffics, all traffic sources always have a packet to send and always send a 1000-bytes packet as soon as the congestion control window allows them to do so. The receiver immediately sends an acknowledge (ACK) packet when it receives a data packet. For both RED and Fuzzy mechanism, the max_th, min_th and wq were set to 80%buffer size, 20%buffer size and 0.002, respectively.

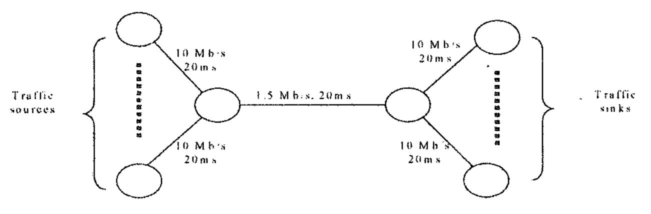


Figure 11: The simulation topology

In Figure 12, the packet loss probability of all mechanisms is 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. This figure shows the packet loss of our proposed fuzzy controller is 2% less than those of RED1 and RED0.01.

To evaluate the performance of proposed fuzzy controller under different congestion density, we performed more simulations. In this case, four traffic sources start at time 0 and at time 50, sixteen new traffic sources start to send the packets. The packet loss rate of all mechanisms is shown in Figure 13. This figure shows that for all mechanisms at time 50, the packet loss rate is increased. It can be seen that the packet loss rate of the fuzzy controller is less than those of RED1 and RED0.01.

In the next simulation, we evaluate the performance of all mechanisms with a decrease in congestion. In this case, twenty traffic sources start at time 0. At time 50, sixteen traffic sources stop the sending of packets. Figure 14 shows the packet loss rate of RED1, RED0.01 and Fuzzy. As can be seen in this figure, for all mechanisms at time 50, the packet loss rate is decreased. It is clear that the packet loss rate of the proposed fuzzy controller is less than those of RED1 and RED0.01.

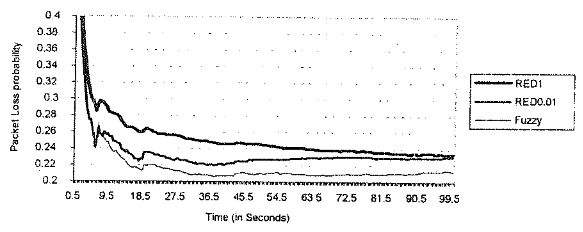


Figure 12: Packet loss rate of RED1, RED0.01 and Fuzzy

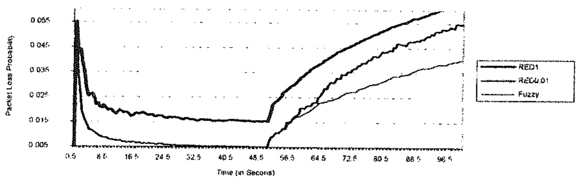


Figure 13: Packet loss rate of RED1, RED0.01 and Fuzzy (with an increase in congestion)

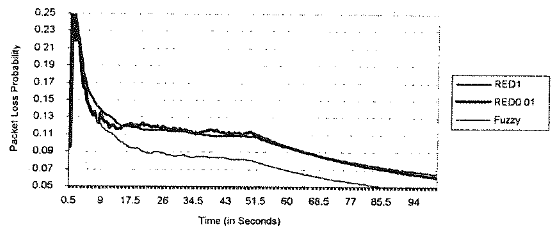


Figure 14: Packet loss rate of RED1, RED0.01 and Fuzzy (with a decrease in congestion)

C. Fuzzy BLUE (FBLUE)

To resolve the problems of traditional RED mechanism, the BLUE [34] algorithm was proposed. The main idea of BLUE is to use different parameters such as packet loss and link utilization to detect congestion and adjust the rate of packet dropping/marking. BLUE uses single probability, P_m, which is used to mark or drop packets when they are enqueued. If the queue is continually dropping packets due to overflow, P_m is increased. If the link is underutilized, the probability is decreased. The amount of increase and decrease are d1 and *d2*, respectively. The rate of updating P_m is *1/freeze*_time. This allows P_m to take effect before the next update. The probability is also updated every time the queue length exceeds a threshold *L*. In other words, it does not let the queue become full. This allows space to be reserved in the queue for occasional bursts. The important features of the BLUE are the minimal amount of buffer space, the reduction of end-to-end delay, the high link utilization and a very low rate of packet loss.

The proposed Fuzzy BLUE is an extension to the traditional BLUE mechanism. The input linguistic variables are the packet loss (pLoss) and the normalized queue length (nqLen). The output linguistic variable is the drop probability (P_m). In certain periods of time, the amount of these two variables are monitored and fed to the Fuzzy BLUE. These values are then fuzzified using linguistic variables and the fuzzifier procedure. Referring to the rule base, rules which are fired according to these fuzzified values (i.e., rules which have the same 'if-part') are determined. To select the set of fired rules, we used min-max combination method which defines the zone which the output value belongs to.

Simulation Results

To compare the performance of the proposed Fuzzy BLUE mechanism with that of the traditional RED and BLUE mechanisms, we performed many simulation trials the maximum buffer size, w_q and $\max_p$ were set to 100 packets, 0.002 and 0.1, respectively. As explained in [34], d1 is an order of magnitude larger than d2. Furthermore, the freeze_time is varied from 10 ms to 100 ms corresponding to the typical value of Round Trip Time (RTT) in the Internet. The other parameters of RED, BLUE and FAQM mechanisms are given in table 1. We used a dumbbell network topology with the link capacity equal to 5 Mb/s. In this case, 6 traffic sources are directly connected to an IP router. Three traffic sources generate AF11 (GREEN) packets, two traffic sources generate AF12 (YELLOW) packets and one traffic source generate AF13 (Red) packets. All traffic sources are data sources.

TABLE 1
THE PERFORMANCE OF RED, BLUE AND FAQM MECHANISMS

AQM Mechanism	Parameter	High priority (GREEN) packets	Middle priority (YELLOW) packets	Low priority (Red) packets
RED	$\min_{th}$	20 packets	30 packets	40 packets
	Max_{th}	80 packets	70 packets	60 packets
BLUE	d1	0.002	0.002	0.002
	d2	0.0002	0.0002	0.0002
	Freez_time	10 ms	10 ms	10 ms
	Queue limit	100 packets	80 packets	60 packets
FAQM	Freeze_time	10 ms	10 ms	10 ms
	Queue limit	100 packets	80 packets	60 packets

In Figure 15, for three different AQM mechanisms and for GREEN and YELLOW packets, the packet loss ratio, the link utilization and the delay are plotted versus simulation time. As shown in these figures, it can be seen that, the proposed FAQM mechanism outperforms the traditional RED and BLUE mechanism. Note that in this simulation the generated traffics are directly entered to the output buffer and the meter is bypassed.

Based on results shown in Figure 15(a,c,e,g), the proposed FAQM in compared with both the BLUE and RED mechanisms has a better packet loss performance. It can be easily observed that at the end of simulation, the total packet loss of the RED, BLUE and FAQM mechanisms are equal to 14.5%, 11.5% and 5 %, respectively. So, in comparison with RED and BLUE mechanisms, the proposed FAQM has more than 7% less packet loss. This is because our proposed FAQM can tune the drop probability Pm dynamically to achieve a low packet loss rate. As shown in Figure 15(h), in comparison with the RED and BLUE mechanisms, the FAQM has better total channel utilization.

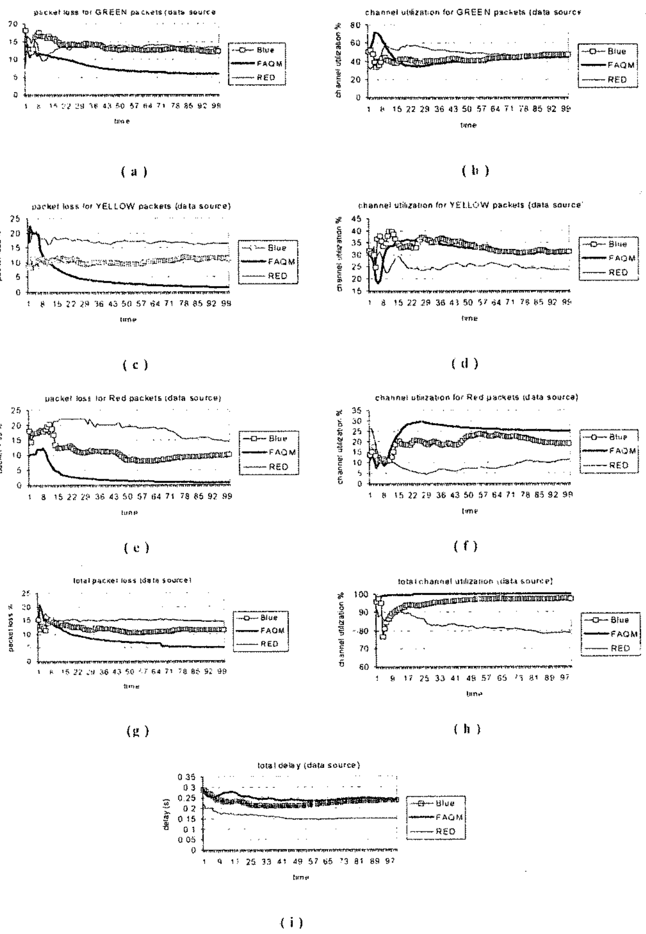


Figure 15: Performance comparison of FAQM, RED and BLUE: (a) packet loss versus simulation time (GREEN packets); (b) channel utilization versus simulation time (GREEN packets); (c) packet loss versus simulation time (YELLOW packets); (d) channel utilization versus simulation time (YELLOW packets); (e) packet loss versus simulation time (Red packets); (f) channel utilization versus simulation time (Red packets); (g) total packet loss versus simulation time; (h) total channel utilization versus simulation time; (i) total delay versus simulation time

As the RED mechanism has the lowest channel utilization, it is expected to have a better delay performance in comparison with BLUE and FAQM mechanisms. In Figure 15(i), the total delay is plotted versus simulation time. It can be seen that the delay performance of proposed FAQM is approximately equal to that of BLUE mechanism.

D. *Fuzzy Scheduling (FS)*

The scheduling algorithms aim at distributing the output channel capacity fairly among the different traffic flows - often taking into account the different quality of service requirements agreed on at admission-time of the flow. The proposed Fuzzy Scheduler is based on the Weighted Fair Queuing (WFQ) [42] mechanism developed by Demers, Keshav, and Shenker in 1989. The idea behind the WFQ algorithm is that for each packet, the WFQ computes the time at which service to the packet would be finished deploying a GPS scheduler. Then, the WFQ scheduler services the packet in the increasing order of their finish times. In other words, the WFQ simulates the GPS on the side and uses the results of this simulation to determine the packets' service order.

The WFQ guarantees the weighted fair share of the output port bandwidth to each service class with a bounded delay. In the WFQ mechanism, queues are served according to their configurable weights that can be changed during the network operation. This allows us to have control over the bandwidth assigned to each service class. By changing this weight, we can control both the packet delay and the drop rate for each queue (class). As we discussed earlier, the WFQ assigns a finish number to each packet that relates inversely the weight of the class to the packet which belongs to. If we increase the weight of the class during the network operation, the finish time of the packets belonging to that class will become smaller and hence will receive service sooner than packets of other service classes. As a result, the queuing delay of these packets becomes smaller. Furthermore, the overflow of the buffer and hence the packet loss will occur less for this service class because the packets in this class receive service faster.

The proposed Fuzzy Scheduler attempts to make a balance between the AF drop and the EF delay using the WFQ scheduler as a base, and then to adjust the weights of the AF and EF queues when necessary. The input variables of the fuzzy controller are the packet loss rate of the AF class (AF_drop) and the delay of the EF class (EF_delay). At regular intervals, the amount of these two variables are monitored and fed to the fuzzy controller. The output of the fuzzy controller is then used to tune the weights of he AF and EF queues.

The output variable of the Fuzzy Scheduler is the amount of variations in the queue weights. This amount will be multiplied by the current weight of the queue and then will be added to the old weight of the queue to form the new weight. The proposed Fuzzy Scheduler has two different sets of rule base, one for the AF class and the other for the EF class. For example, if the drop rate of the AF packets goes high and the delay of the EF packets is low, we can increase the weight of the AF class while maintaining the EF class weight unchanged. Conversely, if the delay of the

EF packet increases (falls in the 'high' zone), then we can increase the weight of the EF class so that EF packets receive service sooner in the next interval and hence, the EF delay is decreased. When there is a need to increase both weights, new weights should be normalized in a way that the sum of weights remains 1.

Simulation Results

To evaluate the performance of Fuzzy Scheduler, we simulated a simple dumbbell network topology with four data sources, two routers, and four destinations. The simulation network topology is shown in Figure 16. The TCP New Reno is used as a congestion control scheme with window size equal to 50 packets.

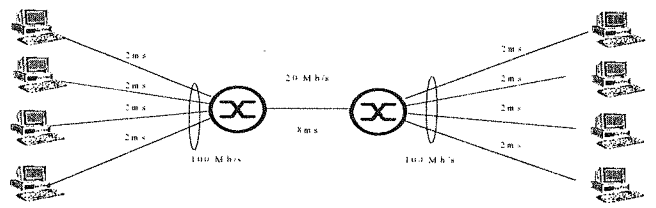


Figure 16: Network topology

As shown in Figure 17, in this topology, sources #1 and #2 produce the AF traffic, and sources #3 and #4 produce the EF traffic. The AQM scheme for the AF class is RED (with $\min_{th}$=0.2B, $\max_{th}$=0.8B, $\max_{p}$=0.1, w_q=0.002) and for the EF class is FIFO. The buffer size (B) was set to 70 packets. We performed the simulation 20 times, each lasts for 100 seconds. The average results of this execution have been shown in the figure.

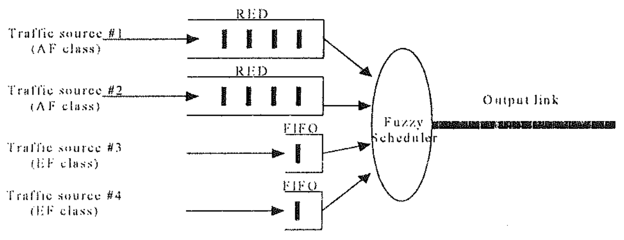
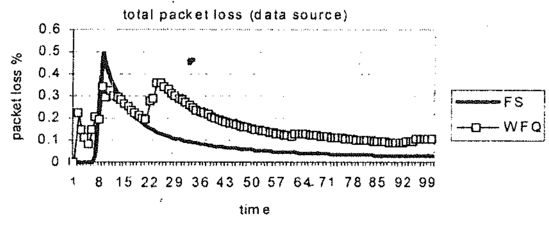

Traffic source #1 (AF class)
Traffic source #2 (AF class)
Traffic source #3 (EF class)
Traffic source #4 (EF class)
RED
RED
FIFO
FIFO
Fuzzy Scheduler
Output link

Figure 17: The queuing model of the network topology used for simulation

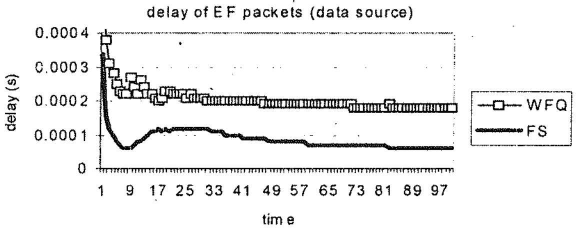
Figure 18 (a) shows the AF drop rate of the proposed FS and the original WFQ mechanism. Obviously, the FS has a lower packet loss ratio than the traditional WFQ mechanism does. In Figure 18 (b), for both schedulers the average queuing delay of the EF packets are plotted versus simulation time. As shown in this figure, it can be seen that the FS has a better delay performance than the traditional WFQ mechanism does. The link utilization of the FS and the WFQ is shown in Figure 18(c). It is clear that the FS has more channel utilization than WFQ mechanism does.

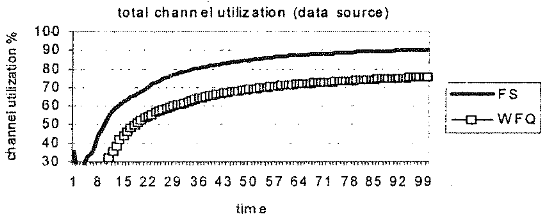
D. Fuzzy TCP

Over the years, several different flavors of TCP have been introduced, such as TCP Tahoe [43], TCP Reno [44], TCP New Reno [45] and TCP Vegas [46].



(a)





(c)

Figure 18: Performance comparison of WFQ and FS: (a) packet loss versus number of sources (AF class); (b) queuing delay versus number of sources (EF class); (c) total channel utilization versus number of sources

The main objective of Fuzzy TCP is to use fuzzy logic capabilities to dynamically tune the TCP congestion window. The proposed Fuzzy TCP is based on TCP Vegas. TCP Vegas was first introduced by Brakmo et al. in [46].It primarily enhances the congestion avoidance and fast retransmission algorithms of TCP Reno [44]. There are the following changes made in TCP Vegas:

- **New retransmission mechanism:** TCP Vegas introduces three changes that affect TCP's (fast) retransmission strategy. First, TCP Vegas measures the RTT for every segment sent. The measurements are based on fine-grained clock values. Using the fine-grained RTT measurements, a timeout period for each segment is computed. When a duplicate acknowledgement (ACK) is received, TCP Vegas checks whether the timeout period has expired. If so, the segment is retransmitted. Second, when a Non-duplicate ACK that is the first or second after a fast retransmission is received, TCP Vegas again checks for the expiration of the timer and may retransmit another segment. Third, in case of multiple segment loss and more than one fast retransmission, the congestion window is reduced only for the first fast retransmission.

- **Congestion avoidance mechanism:** TCP Vegas does not continually increase the congestion window during congestion avoidance. Instead, it tries to detect incipient congestion by comparing the measured throughput to its notion of expected throughput. The congestion window is increased only if these two values are close, that is, if there is enough network capacity so that the expected throughput can actually be achieved. The congestion window is reduced if the measured throughput is considerably lower than the expected throughput; this condition is taken as a sign for incipient congestion.
- **Modified slow-start mechanism:** A similar congestion detection mechanism is applied during slow-start to decide when to change to the congestion avoidance phase. To have valid comparisons of the expected and the actual throughput, the congestion window is allowed to grow only every other RTT.

The congestion avoidance mechanism that TCP Vegas uses is quite different from that of TCP Tahoe or Reno. TCP Reno uses the loss of packets as a signal that there is congestion in the network and has no way of detecting any incipient congestion before packet losses occur. Thus, TCP Reno reacts to congestion rather than attempts to prevent the congestion. TCP Vegas uses the difference between the estimated throughput and the measured throughput as a way of estimating the congestion state of the network.

TCP Vegas sets BaseRTT to the smallest measured Round Trip Time (RTT), and the expected throughput is computed according to following equation:

$$Expected = \frac{cwnd}{BaseRTT} \qquad (5)$$

where *cwnd* is the current window size. With each packet being sent, TCP Vegas records the sending time of the packet by checking the system clock and computes the round trip time by computing the elapsed time before the ACK comes back. It then computes actual throughput using this estimated RTT according to following equation:

$$Actual = \frac{cwnd}{RTT} \qquad (6)$$

Then, TCP Vegas compares Actual to Expected and computes the difference *delta* as below:

$$delta = Actual - Expected \qquad (7)$$

The *delta* is used to adjust the window size. To achieve this, TCP Vegas defines two threshold values, α, β (α<β). If *delta* < α, the window size is increased linearly during the next RTT. If *delta* > β, then TCP Vegas decreases the window size linearly during the next RTT. Otherwise, it leaves the window size unchanged.

As mentioned above, the TCP Vegas tunes the congestion windows linearly. The main problems of TCP Vegas is that it only compare the *delta* with two fixed thresholds and it does not consider the distance between *delta* and two thresholds α,β. The main objective of current study is to use fuzzy logic capabilities to dynamically tune the congestion window. To do this, if the distance between *delta* and α,β is very low, low, high or very high , we can change the window size very low, low, high or very high, respectively.

The proposed fuzzy based TCP congestion controller uses a single-input single-output fuzzy controller. The input parameter to the controller is *delta*. The output of fuzzy controller, *Adjust*, is used to tune the value of congestion window accurately. The input and output Fuzzy sets of the linguistic variables are shown in Figure 18 and Figure 19, respectively.

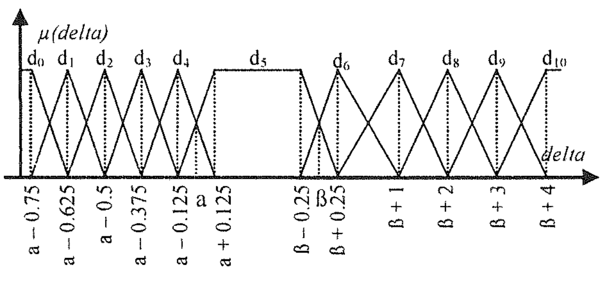


Figure 18: The fuzzy sets of *delta*

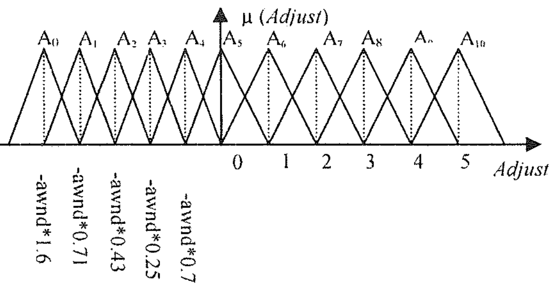


Figure 19: The fuzzy sets of *Adjus*

In Figure 19, the awnd parameter represents the average of the send window size. The output of fuzzy controller (*Adjust*) is added to the current window size and the new window size is calculated. The proposed fuzzy controller uses the following rules in its rule base:

Rule 0: if delta is d_0 then Adjust A_{10}
Rule 1: if delta is d_1 then Adjust A_9
Rule 2: if delta is d_2 then Adjust A_8
Rule 3: if delta is d_3 then Adjust A_7
Rule 4: if delta is d_4 then Adjust A_6
Rule 5: if delta is d_5 then Adjust A_5
Rule 6: if delta is d_6 then Adjust A_4
Rule 7: if delta is d_7 then Adjust A_3
Rule 8: if delta is d_8 then Adjust A_2

Rule 9: if delta is d_9 then Adjust A_1
Rule 10: if delta is d_{10} then Adjust A_0

Using the single tone fuzzifier, product inference engine and center of average defuzzifier, the final output of fuzzy controller is calculated as below:

$$Adjust = f(delta) = \frac{\sum_{l=0}^{10} \overline{A}^l . \mu_{d^l}(delta)}{\sum_{l=0}^{10} \mu_{d^l}(delta)} \qquad (8)$$

where l is the rule number and $\overline{A}^l$ is the center of fuzzy set used in the Then part of l th rule.

Simulation Results

To evaluate the performance of the proposed fuzzy controller, we used the ns-2 simulator. The proposed model which is called Fuzzy Vegas was implemented in the ns-2 simulator. Figure 20, shows the structure of first simulated network.

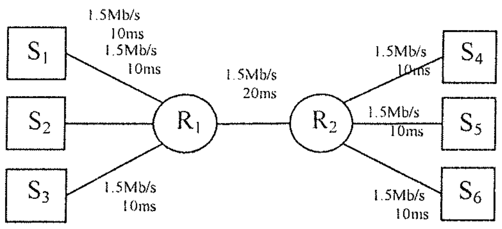


Figure 20: The simulated network

In the simulation scenario, the connection between (S_1, S4) is TCP and for (S_2, S_5), (S_3, S_6) the UDP protocol is used. The S_1, S_2 and S_3 transmit data and the S_4, S_5, S_6 receive the data packets. All queues in the nodes are based on Drop Tail strategy. An FTP traffic source is attached to node S_1 and Constant Bit Rate (CBR) traffic with 1.5Mb/s sending rate attached to S_2, S_3. S_2, S_3 transmit data at t=5s and are stopped respectively at t=10s and t=30s. S_3 restarts to send packets at t=50s and it is stopped after 10s, S_3 restarts to send at t=60s and stops at t=70s the same as S_2. The node S_1 starts at t=0s and is stopped at the end of the simulation. In this scenario S1 is once simulated with TCP Vegas agent and another time with the Fuzzy Vegas. The network throughput for connection between S_1 and S_4 is plotted versus the simulation time in Figure 21. In Figure 22, the variation of *cwnd* is plotted versus the simulation time for both of them. As it can be seen, the proposed Fuzzy Vegas can tune the *cwnd* in a much better form than TCP Vegas does. In table 2, for both TCP Vegas and Fuzzy Vegas, the number of duplicate ACK packets received by the sender has been shown. Based on the results shown in Figures 21, 22 and Table 2, it can easily be seen that the proposed Fuzzy Vegas has a better performance than the traditional TCP Vegas.

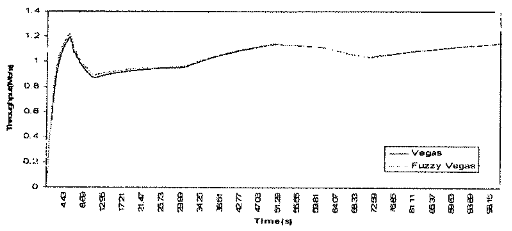


Figure 21: Throughput versus simulation time (Drop tail strategy at R_1)

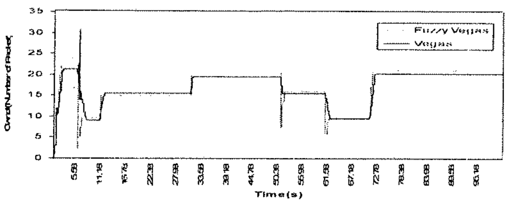


Figure 22: The *cwnd* versus simulation time (Drop tail strategy at R_1)

We repeated the above scenario on the simulated network shown in Figure 20, with queue based on RED strategy for R_1. The simulation results are given in Figure 23,24 and Table 2.

TABLE 2

TOTAL NUMBER OF DUPLICATED ACKNOWLEDGE RECEIVED BY THE SENDER

	Fuzzy Vegas	Vegas
Drop Tail	36	282
RED	2153	2217

Based on the results shown in Figures 21-24 and Table 2, it can be easily seen that the proposed Fuzzy Vegas has a better performance compared with the traditional TCP Vegas.

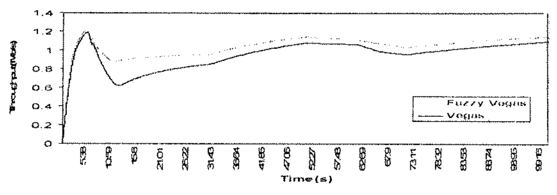


Figure 23: Throughput versus simulation time (RED strategy at R_1).

The Structure of the second simulation is similar to first simulation with some changes in link capacity, link delays and also the length of the scenario. The structure of the second simulation network is shown in Figure 25. All queues in the nodes are based on Drop Tail strategy.

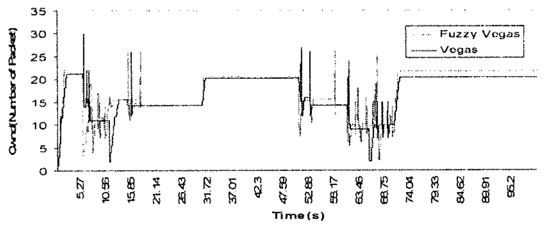


Figure 24: The *cwnd* versus simulation time (RED strategy at R_1)

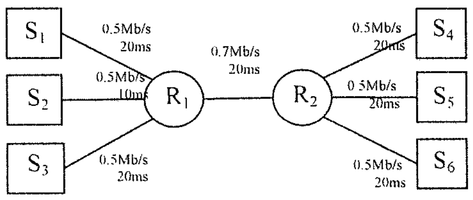


Figure 25: The second simulated network

S2 and S3 are again UDP sources and S5, S6 are sinks, respectively. The Connection between S1 and S4 is TCP. S2 and S3 transmit data at constant bit rate of 0.5Mb/s. S2 starts sending packets at t=0s and doesn't stop before the end of the simulation. S3 starts at t=15 and stops at t=25. S1 is a source of FTP traffic, it starts sending packets at t=0s and continues till the end of simulation. S1 was attached to the TCP variants presented in section 1 and the throughput of the TCP connection between S1 and S2 was measured each time. The measured throughput is plotted versus simulation time in Figure 26.

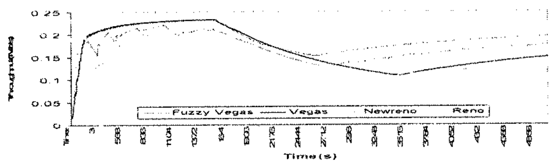


Figure 26: Throughput versus simulation time

Table 3 shows the total duplicates acknowledge received by sender. The Final simulation was conducted to evaluate the influence of increasing the number of nodes on the simulated network on the performance of the proposed model. Here we increase the total number of nodes in the simulated network to 20.

Figure 27 shows the details of the simulated network. All queues in the nodes are based on Drop Tail strategy again. S_1 and S_{11} are TCP connections and connection of other pairs of nodes are UDP, S_1 starts sending FTP traffic at t=30s and stops at t=150s, others start sending packets at constant bit rate at t=5s, t=10s ... t=45s respectively, and stop accordingly at t=85s, t=90s, t=125s. The throughput for connection between S_1 and S_4 is plotted versus simulation time in Figure 28.

TABLE 3

TOTAL NUMBER OF DUPLICATED ACKNOWLEDGE RECEIVED BY SENDER

	Duplicate Acknowledge
TCP Reno	5128
TCP New Reno	12117
TCP Vegas	4032
TCP Fuzzy Vegas	322

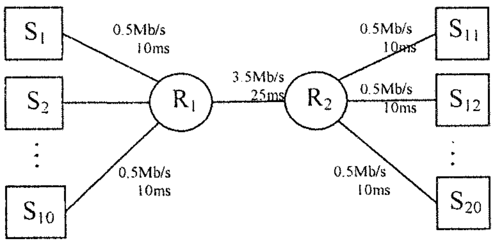


Figure 27: The third simulated network

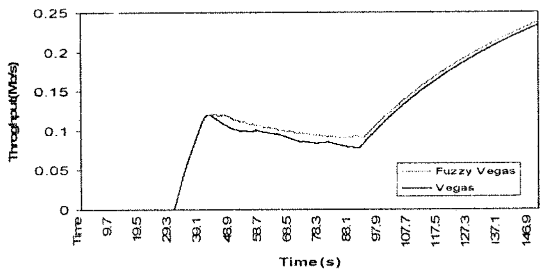


Figure 28: The throughput versus simulation time

IV. CONCLUSION

In this paper we presented the application of fuzzy logic controllers in the different area of networking including: ATM and Internet Protocol. We proposed two fuzzy controller namely FCAC and FUPC where are responsible for connection admission control and usage parameter control, respectively. The application of fuzzy logic in different aspects of differentiated services routers was studied. We proposed Fuzzy RED (a Fuzzy implementation of traditional RED algorithm), Fuzzy BLUE (a fuzzy implementation of traditional BLUE algorithm), Fuzzy srTCM, Fuzzy Scheduling and Fuzzy TCP. Fuzzy congestion window controller achieved better performance by integrating a fuzzy controller and nonlinear adjusting end to end congestion windows in TCP. Based on simulation results it can easily be seen that all proposed fuzzy mechanisms have a better performance in comparison of traditional none fuzzy mechanisms.

REFERENCES

[1] ITU_T Recommendation I.371, "Traffic Control and Congestion Control in BISDN"

[2] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", IETF RFC 1633, June 1994.

[3] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", IETF RFC 2205, September 1997

[4] Mankin, Ed., F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, L. Zhang , "Resource ReSerVation Protocol (RSVP) -- Version 1 Applicability Statement Some Guidelines on Deployment", IETF RFC 2208, September 1997.

[5] T. Li, Y. Rekhter , "A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)", IETF RFC 2430, October 1998.

[6] K. Nichols, V. Jacobson, L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet", IETF RFC 2638, July 1999.

[7] J. Heinanen et al. "Assured Forwarding PHB Group", February 1999, draft-ietf-diffserv-af-06.txt

[8] Jacobson, K. Nichols and K. Poduri, "An Expedited Forwarding PHB", February 1999, draft-ietf-diffserv-phb-ef-02.txt

[9] A.R. Bonde, S. Ghosh, "A comparative study of fuzzy versus fixed threshold for robust queue management in cell-switching networks", IEEE/ACM Trans. Networking 2 (4) (1994) 337±344.

[10] Marcial Porto Fernandez, Aloysio de Castro P. Pedroza and José Ferreira de Rezende, "QoS Provisioning across a DiffServ Domain using Policy-Based Management" , in *(Globecom 2001)*, (San Antonio, USA), Nov. 2001.

[11] M.H.Yaghmaee et al."A fuzzy variable rate leaky bucket for ATM traffic control", in the proceeding of European Congress on Intelligent Techniques and Soft Computing (EUFIT98), Germany, 1998.

[12] M.H.Yaghmaee M.Safavi, M.B.Menhaj, "A novel FLC based approach for ATM traffic control", Elsevier Science Journal on Computer Networks, Vol. 36/5-6, pp.643-658, 2001

[13] M.H.Yaghmaee, M.Safavi, M.B.Menhaj," An intelligent usage parameter controller based on dynamic rate leaky bucket for ATM networks", Elsevier Science Journal on Computer Networks, Vol. 32, pp.17-34, 2000.

[14] M.H.Yaghmaee, M.Safavi, "A Fuzzy Based Three Color Meter/Marker for Diffserv Networks", , The International Journal of Engineering, National Center For Scienific Research,Vol.17,No.3, September 2004.

[15] M.H.Yaghmaee, "A Modified Random Early Detection Algorithm: Fuzzy Logic Based Approach ",to be published in Journal of Communication Network (JCN), Korea,2005

[16] M.H.Yaghmaee, M.B.Menhaj, H.Amintoosi,"A Fuzzy Extension to the BLUE Active Queue Management Algorithm", to be published in the journal of Iranian Association of Electrical and Electronics Engineers, 2005

[17] J.S. Turner, "New direction in communication (or which way in the information age?)", in: Proceedings of the Int. Zurich Sem. Digit. Communication, Zurich, Switzerland, March 1986, pp. 25-32.

[18] M. Butto, E. Cavallero, "Effectiveness of the leaky bucket policing mechanism in ATM networks", IEEE J. Select. Areas Commun. 9(3) (1991) 335-342.

[19] E.P. Rathgeb, "Modeling and performance comparison of policing mechanisms for ATM networks", IEEE J. Select. Areas Commun. 9 (3) (1991) 325-334.

[20] G. Niestegge, 'The leaky bucket policing method in ATM", Int. J. Digital Analog Commun. Sys. 3 (1990) 187-197.

[21] N. Yamanaka, Y. Sato, K. Sato, "Performance limitation of leaky bucket algorithm for usage parameter control and bandwidthallocation method", IEICE Trans. Comm. E75-B (1992) 82-86.

[22] H. Ahmadi, R. Guerin, K. Sohraby, "Analysis of leaky bucket access control mechanism with batch arrival process", in: Proceedings of IEEE Globcom '90, 1990.

[23] A.W. Berger, "Performance analysis of a rate control where tokens and jobs queue", IEEE J. Select. Areas Commun. 9 (2) (1991)165-170.

[24] H.J. Chao, "Design of leaky bucket access control schemes in ATM networks", in: Proceedings of ICC '91, 1991, pp. 180-187.

[25] T.F. Ibrahim, C.V. Chakravarthy, "The enhanced exponentially weighted moving method: a modified enforcement technique for ATM networks", in: Proceedings of ICC '94, 1994, pp. 1379-1381.

[26] Lee J.Y , C.K.Un. , 1993, " PERFORMANCE OF DYNAMIC RATE LEAKY BUCKET ALGORITHM",Electronics letters,19th August 1993,Vol.29,No.17,pp.1560-1561.

[27] Butto M.,Cavallero E.,Tonietti A. ,1991," Effectiveness of the Leaky Bucket Policing Mechanism in ATM Networks", Journal on Selected Areas in Communications,Vol.9,No.3,April 1991,pp.335-342.

[28] J. Heinane, Telia Finland, R. Guerin, "A Single Rate Three Color Marker", IETF RFC 2697, September 1999.

[29] J. Heinane. R. Guerin, "A Two Rate Three Color Marker", IETF RFC 2698, 1999.

[30] W. Fang, "A Time Sliding Window Three Colour Marker (TSWTCM)", IETF RFC 2859, June 2000.

[31] W.Feng, et al, " Adaptive Packet Marking for Providing Differentiated Services in the Internet", University of Mishigan and IBM

[32] S. Floyd and V. Jacobson. "Random Early Detection Gateways for Congestion Avoidance.", IEEE/ACM Transactions on Networking, 1(4):397–413, Aug. 1993.

[33] M.May, J.Bolot, C.Diot, B.Lyles, "Reasons Not to Deploy RED", Proc. of 7th International Workshop on Quality of Service (IWQoS'99), pp. 260-262, June 1999

[34] W.Feng, D.Kandlur, D.Saha, K.Shin, "BLUE: A New Class of Active Queue Management Algorithms" U.Michigan CSE-TR-387-99, April 1999

[35] T.Ott, T.Lakshman, L.Wong, "SRED: Stabilized RED", in proceeding IEEE INFOCOM 1999

[36] 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, November 1997.

[37] W.Feng, D.Kandlur, D.Saha, and K.Shin, " A self Configuring RED Gareway", Infocom, Mar 1999.

[38] S.Floyd, R.Gummadi, S.Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED", Technical Report, 2001

[39] Sanjeewa Athuraliya, Victor H. Li, Steven H. Low, Qinghe Yin, "REM: Active Queue Management",IEEE Network, May/June 2001.

[40] Bartek Wydrowski, Moshe Zukerman, " GREEN: An Active Queue Management Algorithm for a Self Managed Internet"

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

[42] Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. SIGCOMM Symposium on Communications Architectures and Protocols, pages 1-12, Sep 1989, Austin, Texas.

[43] V. Jacobson, "Congestion avoidance and control" In Proceedings of SIGCOMM '88

[44] L. S. Brakmo, S. W. O'Malley and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," IEEE Journal on selected areas in communication, Vol. 13, No. 8, October 1995

[45] M. Allman, V. Paxson and W. and Stevens, "TCP Congestion Control." Request for Comments (Standards Track) RFC 2581, Internet Engineering Task Force, April 1999

[46] S. Floyd, and T. Henderson., "The New Reno Modification to TCP's Fast Recovery Algorithm." Request for Comments (Experimental) RFC 2582, Internet Engineering Task Force, April 1999.

[47] P.Carbonell, Z. P. Jiang, S. S. Panwar, "Fuzzy TCP: A Preliminary Study", Proceedings Of the 15th IFAC World Congress (IFAC 2002), Barcelona, Spain, July 21-26, 2002.