A Novel Fuzzy Logic Base Scheduling Mechanism for Service Differentiation in IP Networks

M. H. Yaghmaee[1] M. Bahekmat[2] G. Khojasteh Toussi[3]

[1] Associate Professor, Computer Department, Faculty of Engineering, Ferdowsi University of Mashhad,
Email: yaghmaee@ieee.org

[2] M.Sc., Electrical Engineering Department, Azad University of Mashhad,
Email: Bahekmat_ss@yahoo.com

[3] M.Sc., Electrical Engineering Department, Azad University of Mashhad,
Email: ghazale_khojaste@yahoo.com

Abstract:

Quality of Service (QoS) refers to a set of rules or techniques that help the network administrators use the available network resources optimally to manage the effects of congestion and to treat the applications according to their needs. The differentiated services architecture (DiffServ) allows providing quality of service to users. The major DiffServ premise is that individual flows with similar QoS requirements can be aggregated in larger traffic sets and identified as classes. Relative service differentiation is a simple and easily deployed approach compared to the absolute differentiation service. In this paper, we use fuzzy logic systems to design a novel algorithm for queue management and scheduling of classified packets in the differentiated service IP networks. The proposed model consists of two parts. The first part of the proposed model is used for absolute differentiated services and tries to optimize QoS parameters and to share sources between different requests fairly. The second part of the proposed fuzzy system is dedicated to relative Diffserv model. As one of the famous existing algorithms for both buffer management and scheduling in the relative differentiated service is Jobs algorithm, in the second parts of the proposed system, we modify the traditional Jobs algorithm and proposed a fuzzy based modification of existing Jobs algorithm. The proposed algorithm uses different fuzzy logic controllers to differentiate the delay of traffic classes. Simulation results confirm that the proposed fuzzy system, can provide better delay differentiated than the traditional algorithms.

Keywords: Quality of Service (QoS), Proportional Differentiated Services, Scheduling Mechanisms, Buffer Management, Fuzzy system

Submission date: June 6, 2007

Acceptance date: Apr. 20, 2009

Corresponding author: M. H. Yaghmaee

Address of corresponding author: Computer Department, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

1. Introduction

In the last few years, the growth of the Internet and the use of new services such as e-business, voice over IP (VoIP) [1] and multimedia applications has risen the need to support QoS requirements and to accommodate different service levels.The primary goal of QoS is to provide priority including dedicated bandwidth, controlled jitter and latency (required by some real-time and interactive traffic), and improved loss characteristics. Also important is making sure that providing priority for one or more flows does not make other flows fail [2]. The differentiated services architecture (DiffServ) [3] allows providing quality of service to users. The major DiffServ premise is that individual flows with similar QoS requirements can be aggregated in larger traffic sets and identified as classes. All packets in each traffic class, receive the same 'forwarding behavior' in routers [4]. Two directions exist in the DiffServ architecture: the absolute and the relative. In absolute DiffServ [5], an admission control scheme is used [6] to provide QoS guarantees as absolute bounds of specific QoS parameters such as bandwidth, packet transfer delay, packet loss rate, or packet delay variation (jitter). A connection request is rejected if sufficient resources are not available in the network so as to provide the desirable assurances. End to end performance requires passive or active monitoring procedures along a specific connection before its establishment and throughout its lifetime. Thus, for any admitted user the appropriate resources are reserved and the performance level of the connection is assured [7]. The relative DiffServ model [8] provides QoS guarantees per class in reference to guarantees given to other classes. The only assurance from the network is that higher classes receive better service treatment than lower classes. QoS parameter values for a connection depend on the current network load since there is no admission control and resource reservation mechanism. Relative service differentiation is a simple and easily deployed approach compared to the absolute differentiation service [7]. Proposals for relative per class DiffServ QoS define service differentiation qualitatively [9-10], in terms that higher classes receive lower delays and losses from lower classes. Specifically research effort has focused on a qualitative relative differentiation scheme named proportional DiffServ [11-12], which controls the ratios of delays or loss rates of successive priority classes in order to be constant.

In the following paragraph, a generic description of the proportional differentiation model as described in [11] is given. Suppose that $\bar{q}_i(t,t+\tau)$ is a performance measure for class i in the time interval $(t,t+\tau)$, where $\tau > 0$ is the monitoring timescale. The proportional differentiation model imposes constraints of the following form for all pairs of classes and for all time intervals $(t,t+\tau)$ in which both $\bar{q}_i(t,t+\tau)$ and $\bar{q}_j(t,t+\tau)$ are defined, the following equation is satisfied:

$$\frac{\bar{q}_i(t,t+\tau)}{\bar{q}_j(t,t+\tau)} = \frac{c_i}{c_j} \qquad (1)$$

where $c_1 < c_2 < \ldots < c_N$ are the generic Quality Differentiation Parameters (QDPs). The basic idea is that, even though the actual quality level of each class will vary with the class loads, the quality ratio between classes will remain fixed and controllable by the network operator, independent of the class loads [8]. The proportional differentiation model can be applied in three contexts, proportional delay differentiation [11], proportional loss rate differentiation [12] and proportional jitter differentiation model [13-15]. Several scheduler and dropper are presented about proportional differentiated services [11-21]. Jobs[22] is the famous algorithm in proportional Diffserv model that considers scheduling and buffer management (dropping) together in a single step. Jobs algorithm, proportionally differentiates class of service based on two parameters, delay and loss rate.

In this paper, we use fuzzy logic systems to design a novel algorithm for queue management and scheduling of classified packets in the differentiated service IP networks. As shown in figure 1, the proposed model consists of two parts. The first part of the proposed model is used for absolute differentiated services and tries to optimize QoS parameters and to share sources between different requests fairly. The second part of the proposed fuzzy system is dedicated to relative Diffserv model. As one of the famous existing algorithms for both buffer management and scheduling in the relative differentiated service is Jobs algorithm, in the second parts of the proposed system, we modify the traditional Jobs algorithm and proposed a fuzzy based modification of existing Jobs algorithm. The proposed algorithm uses different fuzzy logic controllers to differentiate the delay of traffic classes. With the expansion of Internet traffic and its diversified service requests, the focus is on requirement of new traffic used on the Internet. In the proposed fuzzy controller for absolute Diffserv model, scheduling packets are based on jitter. The proposed fuzzy controller for relative Diffserv model, proportionally differentiates classes based on jitter and loss rate. Because of jitter is very close to delay, result in this method shows that besides differentiate traffic classes that are proportionate to jitter Differentiation Parameters (JDP) and Loss rate Differentiation Parameters (LDP) , these classes can be separated based on delay as well.

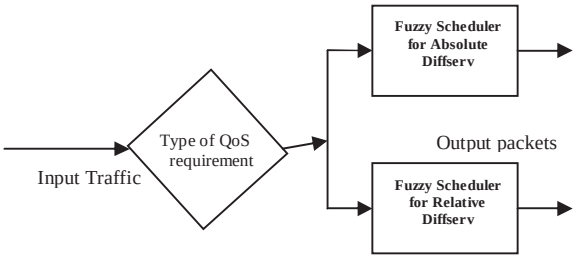


Fig. 1: The structure of proposed fuzzy scheduling mechanism

The reminder of this paper is organized as follows. In section 2, we explain the property of fuzzy systems. In section 3, the proposed fuzzy controller for absolute Diffserv scheduler is described. In section 4 the proposed fuzzy implementation of Jobs algorithm which is used for relative Diffserv model is described. Finally section 5, concludes the paper.

2. A Brief Introduction to Fuzzy Logic Controllers

Fuzzy logic controllers are used to compute values of action variables from observation of state variables of the process under control. Fuzzy logic is very similar to human thinking and natural language. It provides an effective means of capturing the approximate, inexact nature of the real world. The goal of fuzzy logic controller is to put human knowledge into engineering systems. In this paper by using the fuzzy logic capabilities, we develop a fuzzy extension to the traditional Jobs algorithm. In the proposed algorithm, the service rate of all traffic classes is tuned so that both the absolute delay and relative delay constraints are satisfied. As the proposed mechanism, uses the fuzzy controller, we first explain the properties of fuzzy controllers. A fuzzy controller consists of four major parts including: fuzzifier, Inference engine, fuzzy rule base and defuzzifier. 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 a fuzzy controller, 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 a fuzzy controller with n inputs including $x_1, x_2, ..., x_n$ and one output $y \in R$ .The input vector X is defined as: $X = (x_1, x_2, ..., x_n)^T \in R^n$ .Furthermore, suppose the rule base consists of M rules with the following general form:

Rule 1: if x_1 is $A_1^{\,1}$ and x_2 is $A_2^{\,1}$ . . . and x_n is $A_n^{\,1}$ then y is B^1

Rule 2: if x_1 is $A_1^{\,2}$ and x_2 is $A_2^{\,2}$ . . . and x_n is $A_n^{\,2}$ then y is B^2

. . .

Rule M : if x_1 is $A_1^{\,M}$ and x_2 is $A_2^{\,M}$ . . . and x_n is $A_n^{\,M}$ then y is B^M

where in the ith rule, $A_j^{\,i}$ and B^i ($i = 1,2,...,M$; $j = 1,2,...,n$) are fuzzy sets of linguistic variable x_j and y, respectively. In [23] 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:

$$f(X) = \frac{\sum_{l=1}^{M} \bar{y}^l (\min_{j=1}^{n} \mu_{A_j^l}(x_j))}{\sum_{l=1}^{M} (\min_{j=1}^{n} \mu_{A_j^l}(x_j))} \qquad (2)$$

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

3. The Proposed Fuzzy Absolute Diffserv Scheduler

In the first part of the proposed system, we implemented a fuzzy scheduler illustrated in figure 2. A packet classifier classifies incoming packets, puts them in different queues with different priorities. This classification is done based on the level of service they demand from the network. We assigned four different queues, *Q1, Q2, Q3* and *Q4* where *Q1* has the most priority and *Q4* the least. Each queue has a length, *'$QL_i(k)$'*, at any time and an average expiry time of the packets in the queue, *$ET_i(k)$*, where '*i*' is the queue index and '*k*' is the time. These two parameters are used in the scheduling strategy.

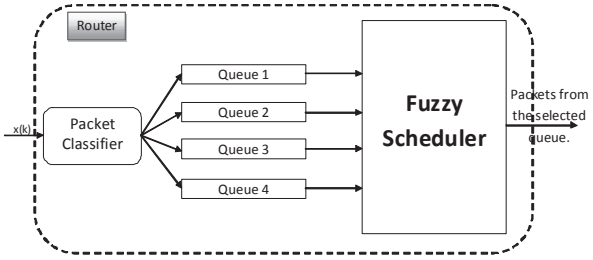


Fig. 2: The location of proposed fuzzy scheduling mechanism in an IP router

The queue lengths change within the time. Assume that $x_i(k)$ is the number of packets added to Q_i at time k. Now, the queue lengths update with the following formula as time passes:

$$QL_i(k+1) = QL_i(k) + x_i(k) - P_i(k) \tag{3}$$

where, if C is the bandwidth capacity we have:

$$P_i(k) = \begin{cases} C & Q_i \text{ selected in time 'k'} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The average expiry time of the packets in the queue is relative to the TTL (Time to Live). TTL is set a default value of 64 seconds. If the packet suffers excessive delays and undergoes multi-hop, its TTL falls to zero. As a result of this, the packet is dropped. If this variable is used as an input to the scheduler for finding the priority index, a packet with a very low TTL value is given the highest priority. The Expiry time is updated using the following procedure; at each time tick, when adding a number of packets to a queue regarding '*$x_i(k)$*' the average expiry time is updated as what follows:

$$ET_i(k+1) = \frac{ET_i(k) * QL_i(k) + TTLx_i(k+1)}{QL_i(k+1)} \tag{5}$$

$TTLx_i(k)$' is the sum of TTLs in '*$x_i(k)$*'. And when removing a number of packets from each queue the average expiry time will be:

$$ET_i(k+1) = \frac{ET_i(k) * QL_i(k) - TTLy_i(k+1)}{QL_i(k+1)} \tag{6}$$

The variable '*$TTLy_i(k)$*' is the sum of the TTL of the packets removed from Q_i at time '*k*'. Also, when time passes we need to update average expiry time of the queues. As soon as a one TTL changes in a packet we update the average expiry time for the queue that packet belongs to. If '*$N_i(k)$*' is the number of packets decreased one unit from their TTLs in time '*k*' from '*i*'th queue, the new average expiry time is:

$$ET_i(k+1) = \frac{ET_i(k) * QL_i(k) - N_i(k)}{QL_i(k+1)} \tag{7}$$

As far as the TTLs do not change rapidly, this equation will not be used frequently.

3.1. Fuzzy Inference

After knowing the network behaviour, we can design the fuzzy inference engine used as a scheduler. In our model, we have assumed to have four queues. For each queue we define two membership functions, one for the queue length and one for the average expiry time of the packets in the queue.We should design a scheduling algorithm that takes into account the queue lengths and their expiry times together with the queue priorities. If the queue lengths are equal and the average expiry times do not significantly vary, it should choose the one with the most priority. Otherwise, it should make a fair, reasonable decision regarding the expiry time and the queue length. Figure 3 illustrates our fuzzy scheduler. It uses a Mamdani fuzzy model [24] with eight inputs and a single output.

In Figure 3, the eight inputs of the fuzzy system are the queue levels QL_1, QL_2, QL_3 and QL_4 and the average expiry times ET_1, ET_2, ET_3 and ET_4. The output is the scheduled queue Q_s, which has four types each corresponding to a separate queue. We assume two fuzzy

membership functions for Queue lengths: small (s), and large (L), and two for the expiry times: low (L) and high (H). All are defined using triangular membership functions. Parameter values in these membership functions must be selected for best performance.

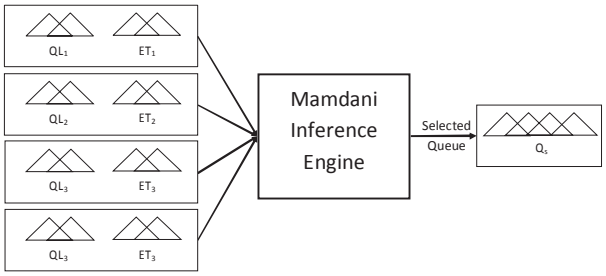


Fig. 3: Proposed Fuzzy Inference System

The range of the horizontal axis in each antecedent membership function is from zero to the maximum queue length for QLs and from zero to maximum TTL for expiry times. With these membership functions specification, we can design the fuzzy rules as the following. We have four variables with three types of membership function and four variables with two types of membership function. So, totally we have $2\times2\times2\times2\times2\times2\times2\times2 = 2^8 = 256$ rules in the knowledge base.

Some of the rules are showed in table 1. As discussed before, the first thing to notice when designing the rule-base is the level of the queue being full. Because, the fuller the queues are the more packets will be dropped. Second, when considering more or less equal queue length the queue with more average expiry time should be selected and after when no significant difference is found in queue levels and average expiry time, the queue priorities are taken into account.

Defuzzification is the final computation stage in the fuzzy scheduling algorithm, in which a certain queue type is determined. We use Centroid of Area (COA) [24], the most popular defuzzifier. Defuzzification yields a representative crisp value from a fuzzy set which is generally not an integer number in this case. Since queue type is an integer 1, 2, 3 or 4, we assign a defuzzified integer with a range of [1,4] using:

$$\text{Selected Queue} = \begin{cases} 1 & z \le 1.5 \\ 2 & 1.5 < z \le 2.5 \\ 3 & 2.5 < z \le 3.5 \\ 4 & 3.5 > z \end{cases} \qquad (8)$$

Table 1: Part of the rule-base designed for the fuzzy scheduler

#	QL_1	QL_2	QL_3	QL_4	ET_1	ET_2	ET_3	ET_4	Q_s
1	S	S	S	S	H	H	H	H	1
2	S	S	S	S	H	H	H	L	4
3	S	S	S	S	H	H	L	H	3
4	S	S	S	S	H	L	H	H	2
…	…	…	…	…	…	…	…	…	…
16	S	S	S	L	H	H	H	H	4
17	S	S	S	L	H	H	H	L	4
18	S	S	S	L	H	H	L	H	4
…	…	…	…	…	…	…	…	…	…
161	L	S	L	S	H	H	H	H	1
162	L	S	L	S	H	H	H	L	1
…	…	…	…	…	…	…	…	…	…
228	L	L	L	S	H	H	L	L	1
…	…	…	…	…	…	…	…	…	…
241	L	L	L	L	H	H	H	H	1
242	L	L	L	L	H	H	H	L	4
…	…	…	…	…	…	…	…	…	…
247	L	L	L	L	H	L	L	H	2
…	…	…	…	…	…	…	…	…	…
255	L	L	L	L	L	L	L	H	1
256	L	L	L	L	L	L	L	L	1

3.2. Performance Evaluation

In this subsection, using computer simulation we evaluate the performance of the proposed fuzzy scheduler. For this purpose we simulated an IP router shown in figure 2. The network environment for our simulations is as follows: four queue types in a router with different priorities and with maximum queue capacity, Q1, Q2, Q3 and Q4, respectively. A link bandwidth with 200 Kb/sec, and a total queue length of 2200 packets. In these simulations, we assume constant-length packets of 256 bytes (i.e. almost 100 packets could be transferred per second). The packet arrivals for each queue type are assumed to have a

Poisson distribution with four different Poisson parameters, λ = 110, 95, 80, and 65, respectively. This means that the queues with more priority receive packets with less probability during the time. For a simple case we just assigned the queue lengths as Q1 = Q2 = 500 and Q3 = Q4 = 600. We used Matlab 7.2.0.232 (R2006a). Figure 4 shows a sample of the input data '$x_i(k)$' fed into the system, created randomly with the mentioned Poisson parameters. TTLs of the packets are set randomly at the time they enter to an integer number in the interval [1, 6 , 4]. Every 2 seconds the packets lay inside the queue with no move one unit is decreased from these values.

We have tested the algorithm and compared with WRR. Simply table 2 shows this comparison. It shows the Maximum number of packets in each queue all over the simulation. Also, it includes the mean and variance of the number of packets within time. What's more, It holds the percentage of packet loss as a result of queue being full (Full Queue Loss%) and the percentage of packet loss because a long delay in the queues and decreasing TTL below zero (Starvation Loss%). The last row of the table shows an average number of seconds the scheduler took to decide which packets from which queue should multiplexed into the output link. All these results are averaged on outcomes of five separate simulation runs. We see that our scheduler is a bit slower. That's because of the big knowledge-base we constructed for the fuzzy inference engine. But losing this point we achieved a fair scheduler, which decides the best decision and allocates recourses for the packets the best it can. This is what the QoS principles are seeking.

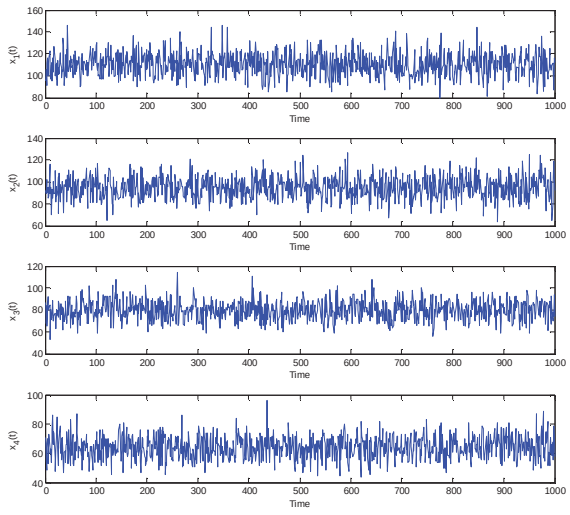


Fig.4: Packet arrival time for all queues

Table 2: Simulation Results

	WRR			
	Q1	Q2	Q3	Q4
Max QL	471.2	499.2	598.3	599.9
Mean QL	165.7	281.7	310.2	410
Variance	210.5	245.1	291.8	298.8
Full Queue Loss%	7.44%			
Starvation Loss%	5.7%			
Average Delay	0.65			
	Fuzzy Scheduler			
	Q1	Q2	Q3	Q4
Max QL	351.5	362.7	398.7	401.5
Mean QL	238.1	249.9	301.8	299
Variance	125.2	137	154.5	159.1
Full Queue Loss%	0.97%			
Starvation Loss%	1.01%			
Average Delay	2.08			

4. Proposed Fuzzy Relative Diffserv Scheduler

In this section we introduce the proposed fuzzy scheduler for relative Diffserv which is fuzzy implementation of Jobs algorithm. The proposed algorithm is called Fuzzy Jobs [25]. The main objective of the proposed Fuzzy Jobs is to enhance the delay differentiation of the Jobs algorithm. The proposed Fuzzy Jobs uses n fuzzy controllers (which n is the number of traffic classes). All fuzzy controllers consist of singleton fuzzifier, minimum inference engine and center average defuzzzifier. In the proposed fuzzy system we use n-1 two-input-single-output fuzzy controllers and one single-input-single-output fuzzy controller. The inputs of all fuzzy controllers are as below:

- Input signal e_1 : This input is an array with n members. Each member of this array belongs to a traffic class. For each traffic class i, the input variable $e_1[i]$ which represents the difference between actual serviced packets and expected service packets, is defined as below:

$$e_1[i] = \left(Rout\_th[i] - Rout[i]\right)/1000000 \qquad (9)$$

For each traffic class i, the Rout[i] represents the number of serviced packets and Rout_th[i] represents the number of expected serviced packets. To reduce the computational complexity, the input signal e1 is scaled by dividing to 1000000. The number 1000000 is selected by trial and error approach. For each traffic class i, if the number of serviced packets is less than the number of expected serviced packets, then $e_1[i]$ will be a positive number. So, to satisfy the defined constraints, the input packets of this class should be serviced faster.

- Input signal e_2 : This input consists of an array with n elements. Some elements of input array are used for traffic classes with absolute constraints while the other are used for relative classes. For each traffic class i with absolute constraints, $e_2[i]$ is defined as below:

$$e_2[i] = \left(delay[i]\right) / ADC[i] \qquad (10)$$

where delay[i] is the delay of packet which is at the head of queue and ADC[i] is the Absolute Delay Constraint of class i. For each class i, if $e_2[i]$ is close to 1 then the packets of this class must be serviced faster than the other classes. Actually $e_2[i]$ represents the error between actual relative delay and the requested relative delay constraint. For relative traffic classes, $e_2[i]$ is defined as below:

$$e_2[i] = \left(delay[i] / delay[i-1]\right) - RDC[i] \qquad (11)$$

where i is the traffic class that RDC (Relative Delay Constraint) has been defined for it, delay[i] represents the delay of packet which is at the head of queue i and RDC[i] is the defined relative delay of class i. Whatever $e_2[i]$ is close to zero, this confirm that for traffic class i the delay differentiation has been satisfied. Suppose we have K classes which relational delay has been defined for them, as the input signal e_2 uses the proportional delay of classes, so for K-1 classes this input signal can be calculated and for the first class it is not possible to calculate the input signal e_2. So for the first class, we use a single-input-single-output fuzzy controller.

- Input signal e_3 : This signal is considered only for the first class which relative delay parameter has been defined for it. The input signal e_3 is calculated similar to e_1 but it has different membership functions. The output of each fuzzy controller determines the service priority of the packet in each traffic class. The packet in a class which has the highest priority is selected to be serviced.

The membership functions of $e_1[1],...,e_1[n]$, $e_2[1],...,e_2[n]$ and e_3 are shown in figure 5.

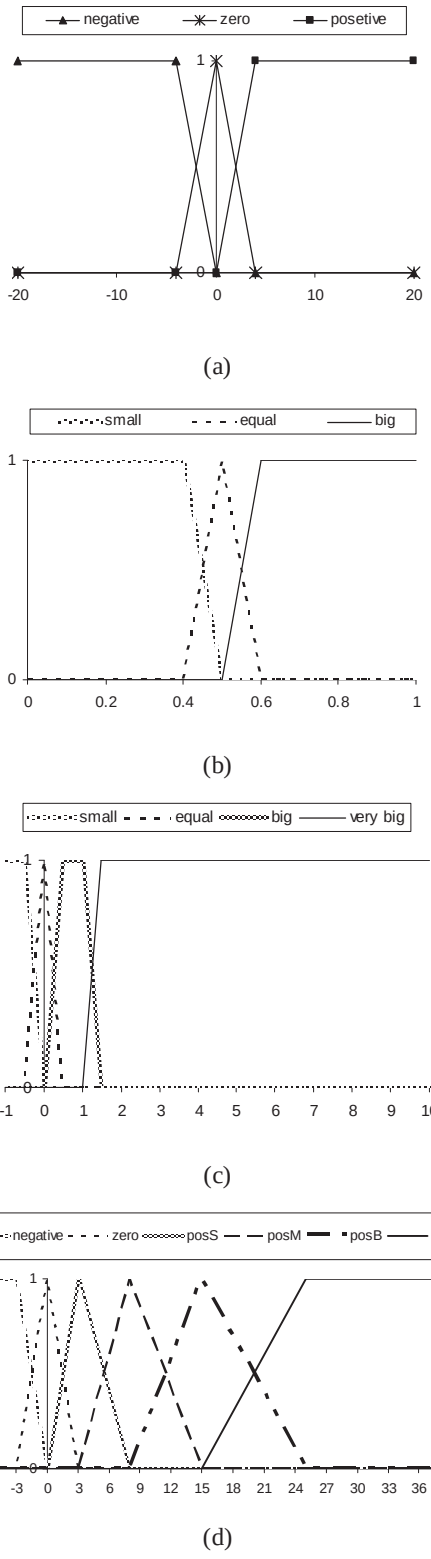


Fig. 5: The membership functions of inputs e_1 , e_2 and e_3 (a) Membership function for $e_1[1],...,e_1[n]$ (b) Membership function for $e_2[k],...,e_2[j]$ (c) Membership function for $e_2[l],...,e_2[m]$ (d) Membership function for e_3

The membership function of the output signal is plotted in figure 6.

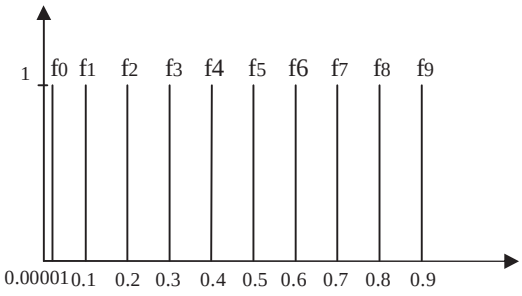


Fig. 6: The output's membership function (p1,…,pn)

The rules of fuzzy controllers are divided to 3 groups. The first group is assigned for classes which use ADC. The fuzzy rules of traffic classes which use RDC are given in group b. Group c is assigned to the first class which uses RDC.

Group a:

Rule 1: If e1[i] is neg and e2[i] is small then priority[i] is f1

Rule 2: If e1[i] is neg and e2[i] is equal then priority[i] is f2

Rule 3: If e1[i] is neg and e2[i] is large then priority[i] is f8

Rule 4: If e1[i] is zero and e2[i] is small then priority[i] is f2

Rule 5: If e1[i] is zero and e2[i] is equal then priority[i] is f3

Rule 6: If e1[i] is zero and e2[i] is large then priority[i] is f9

Rule 7: If e1[i] is pos and e2[i] is small then priority[i] is f3

Rule 8: If e1[i] is pos and e2[i] is equal then priority[i] is f5

Rule 9: If e1[i] is pos and e2[i] is large then priority[i] is f9

Group b:

Rule 10: If e1[i] is neg and e2[i] is small then priority[i] is f0

Rule 11: If e1[i] is neg and e2[i] is equal then priority[i] is f3

Rule 12: If e1[i] is neg and e2[i] is large then priority[i] is f5

Rule 13: If e1[i] is neg and e2[i] is vlarge then priority[i] is f9

Rule 14: If e1[i] is zero and e2[i] is small then priority[i] is f0

Rule 15: If e1[i] is zero and e2[i] is equal then priority[i] is f4

Rule 16: If e1[i] is zero and e2[i] is large then priority[i] is f6

Rule 17: If e1[i] is zero and e2[i] is vlarge then priority[i] is f9

Rule 18: If e1[i] is pos and e2[i] is small then priority[i] is f0

Rule 19: If e1[i] is pos and e2[i] is equal then priority[i] is f5

Rule 20: If e1[i] is pos and e2[i] is large then priority[i] is f8

Rule 21: If e1[i] is pos and e2[i] is vlarge then priority[i] is f9

Group c:

Rule 22: If e1[i] is neg then priority[i] is f4

Rule 23: If e1[i] is zero then priority[i] is f5

Rule 24: If e1[i] is posS then priority[i] is f6

Rule 25: If e1[i] is posM then priority[i] is f7

Rule 26: If e1[i] is posB then priority[i] is f8

Rule 27: If e1[i] is posVB then priority[i] is f9

4.1. Simulation Results

To evaluate the performance of the proposed Fuzzy Jobs, we used the ns2 [26] network simulator. The network topology used in the simulation is shown in figure 7.

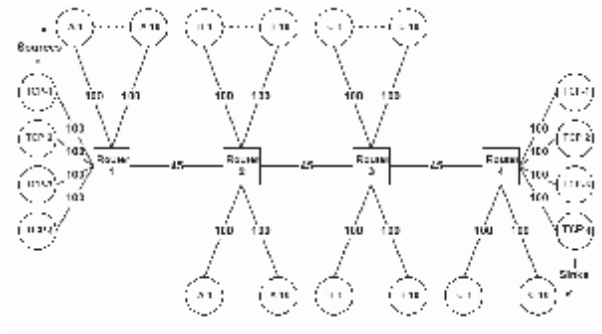


Fig. 7: Network topology used in the simulation

This network topology was used in [22] to evaluate the performance of Jobs algorithm. In this test, just the relative differentiated between the traffic classes is being observed. Network topology consist of four routers that connected by three 45 Mbps links. Sources and sinks are connected to the routers by independent 100 Mbps links. Each 45 Mbps link has a propagation delay of 3 ms, and each 100 Mbps link has a propagation delay of 1 ms. There exist four different traffic classes.

The composition of the traffic mix is given in table 3. Cross-traffic flows (denoted by A-1,…,C-10) start transmitting at time t = 0 s. The flows TCP-1,TCP-2, TCP-3 and TCP-4 start transmitting at time t = 10 s. All flows consist of packets with a fixed size of 500 bytes, and the simulation time is set to 70s. The offered load is asymmetric. Classes1,2,3 and 4 contribute 10%,20%,30% and 40% of the aggregate cross-traffic, respectively.

Table 3: Traffic mix of experiment 1. Traffic mix for flows B-1,..., B-10 and C-1,...C-10 is identical to the traffic mix described here for flows A-1,...,A-10.

Flow	Class	Protocol	Traffic	On	Off
TCP-1	1	TCP	Greedy	N/A	N/A
TCP-2	2	TCP	Greedy	N/A	N/A
TCP-3	3	TCP	Greedy	N/A	N/A
TCP-4	4	TCP	Greedy	N/A	N/A
A-1	1	TCP	Exponential on-of	1000pkts	200ms
A-2,A-3	2	TCP	Exponential on-of	1000pkts	200ms
A-4,A-5,A-6	3	TCP	Exponential on-of	1000pkts	200ms
A-7,A-8,A-9,A-10	4	TCP	Exponential on-of	1000pkts	200ms

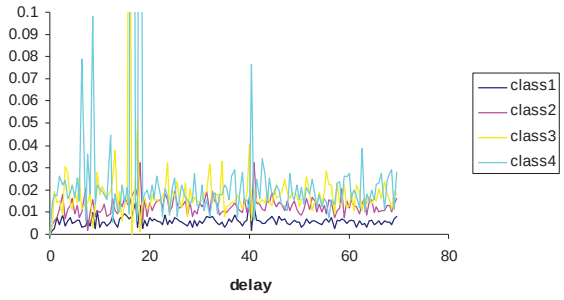
Different experiments were performed. In all experiments we used the same network topology and traffic parameters given in figure 7 and table 3. In the first scenario we only consider the RDC. The service guarantees of traffic classes are as below:

Class-4 Delay ≈ 2 × Class-3 Delay

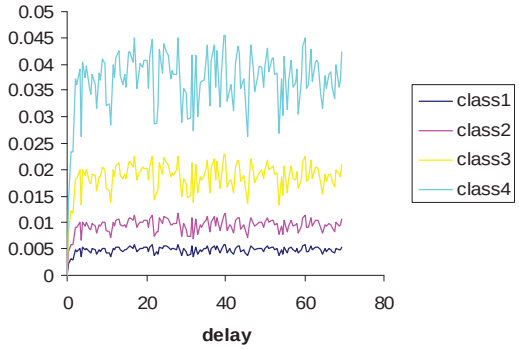
Class-3 Delay ≈ 2 × Class-2 Delay

Class-2 Delay ≈ 2 × Class-1 Delay

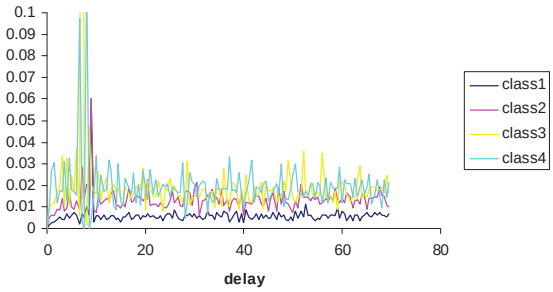
In figure 8, for all routers and for both Jobs and Fuzzy Jobs, the delay of classes is given. As the input traffic load to the Router 4 is less that its output link capacity, so there is not any queuing in this router and the delay is zero.



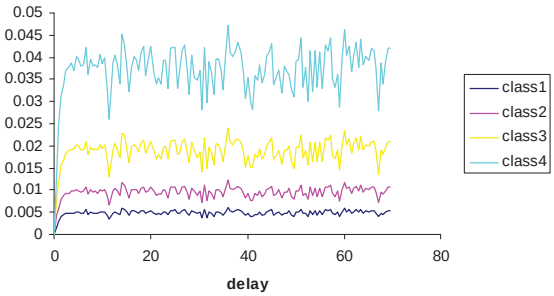
(a)



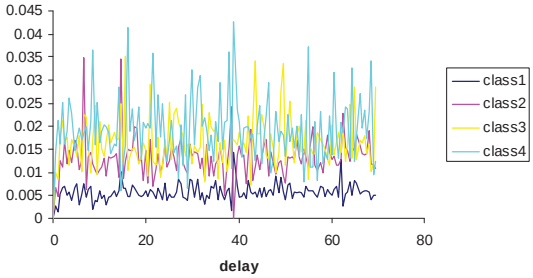
(b)



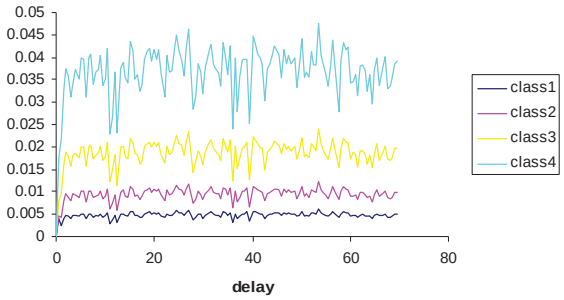
(c)



(d)



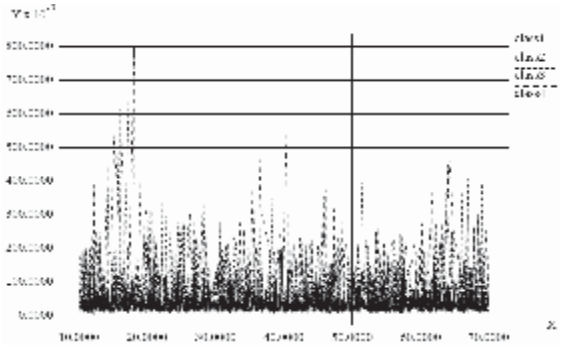
(e)



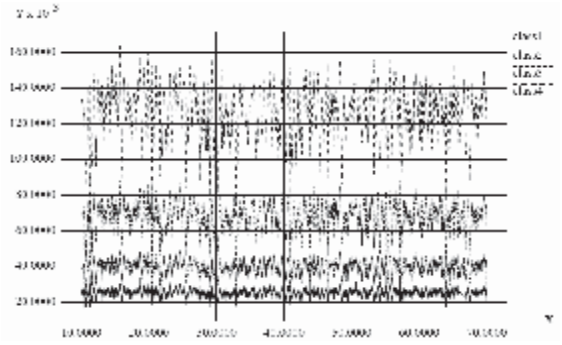
(f)

Fig. 8: Delay of classes for both Jobs and Fuzzy Jobs algorithms (a) Router1- Delay of classes (Jobs) (b) Router1- Delay of classes (Fuzzy Jobs) (c) Router2- Delay of classes (Jobs) (d) Router2- Delay of classes (Fuzzy Jobs) (e) Router3- Delay of classes (Jobs) (f) Router3- Delay of classes (Fuzzy Jobs)

As shown in this figure, it is clear that the proposed Fuzzy Jobs can differentiate the delay of classes better than the traditional Jobs. In figure 9 for both algorithms, the end-to-end delay of flows is shown. By looking at this figure, it can be recognized that the end to end delay of traffic flows in Jobs algorithm is not completely differentiated. But, in the proposed Fuzzy Jobs algorithm this differentiation is perfectly specified and obvious.



(a)



(b)

Fig. 9: The end-to-end delay of flows (a) Jobs algorithm (b) Fuzzy Jobs algorithm

5. Conclusion

In this paper we proposed a fuzzy controller for both absolute and relative differentiated service model. For the absolute Diffserv model, we developed a fuzzy dynamic queue scheduler for class-based queuing in differentiate service routers. A fuzzy rule-base gave us a stable queue scheduler. We simulated a queuing mechanism whit queues with different priorities. Compared to WRR, a well known conventional queue scheduling policy, the proposed fuzzy scheduler yields superior performance in our simulation experiments. The queue levels in the simulated router with our fuzzy scheduler have lower steady-state responses and much lower average queue lengths than WRR. Consequently, the fuzzy based queue scheduler is an efficient dynamic queuing mechanism. The only problem is the delay this scheduler adds to the overall performance of the system. So, it makes it a trade-off between fairness and overall delay. A lot of works in this area is left. Simulations on more realistic network environments are desired. The delay of the system could be simply decreased by an optimal design of the knowledge-base. And also, using better tuned fuzzy membership functions also can enhance the efficiency of the system. For the relative Diffserv model we presented a fuzzy based implementation of traditional Jobs algorithm which is called Fuzzy Jobs. In the proposed Fuzzy Jobs, by using different fuzzy controllers, the service rate of each traffic classes is determined dynamically. The performance of the proposed Fuzzy Jobs was evaluated using computer simulation. Different experiments were performed. All simulation results confirmed that the proposed Fuzzy Jobs has better performance than traditional Jobs.

References

[1] C. Chuah, L. Subramarian, and R. Katz, Furies: A Scalable Framework for Traffic Policing and Admission Control, May 2001, U.C Berkeley Technical Report No. UCB/CSD-01-1144.

[2] Y. Bernet, The Complementary Roles of RSVP and Differentiated Services in the Full-Service QoS Network, *IEEE Communications Magazine*, vol. 38, no. 2, February 2000, pp. 154–162.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W.Weiss, An Architecture for Differentiated Services, IETF RFC 2475, December 1998.

[4] M.K. Leung, J.C. Lui, and D.K. Yau, Characterization and Performance Evaluation for Proportional Delay Differentiated Services Proc. Int'l Conf. Network Protocols, pp. 295-304, Nov. 2000.

[5] B. Teitelbaum, QBone Architecture (v1.0). Internet2 QoS Working Group Draft, http://www.internet2.edu/qos/wg/papers/qbArch/1.0/draft i2-qbone-arch-1.0.html, Aug. 1999

[6] Stoika and H. Zhang, LIRA: An approach for Service Differentiation in the Internet, In Proceedings NOSS-DAV, 1998

[7] Angelos Michalas, Paraskevi Fafali, Malamati Louta, Vassilios Loumos, Proportional Delay Differentiation Employing the CBQ Service Discipline, Telecommunications, 2003. ConTEL 2003.

[8] C. Dovrolis and Parmesh Ramanathan, A Case for Relative Differentiated Services and the Proportional. Differentiation Model. In IEEE Network, 13(5):26-34, September 1999 (special issue on Integrated and Differentiated Services in the Internet).

[9] Odlyzko, Paris Metro Pricing: The Minimalist Differentiated Services Solution, In Proceedings IEEE/IFIP International Workshop on Quality of Service, June 1999.

[10] D. Clark and W. Fang, Explicit allocation of best-effort packet delivery service, In IEEE/ACM Transactions on Networking, Vol. 6 (4), Aug.1998.

[11] C. Dovrolis, D. Stiliadis, and P. Ramanathan, Proportional differentiated services: delay differentiation and packet scheduling, in Proc. ACM SIGCOMM 1999, Cambridge MA, September 1999, pp. 109–120.

[12] C. Dovrolis and P. Ramanathan, Proportional differentiated services, part II: Loss rate differentiation and packet dropping, in *Proc. International Workshop on Quality of Service (IWQoS 2000)*, Pittsburgh, PA, June 2000, pp. 52–61.

[13] T. Ngo-Quynh, H. Karl, A. Wolisz and K. Rebensburg, Relative jitter packet scheduling for Differentiated Services, Proc. Of 9th IFIP working conference on performance modeling and evaluation of ATM&IP networks IFIP ATM&IP, pp. 139-151, 2001.

[14] T. Ngo-Quynh, H. Karl, A. Wolisz and K. Rebensburg, New scheduling algorithm for providing proportional jitter in differentiated services network, Proceedings of IST mobile communication and wireless telecommunications summit, Thessaloniki, Greece, June, 2002.

[15] T. Ngo-Quynh, H. Karl, A. Wolisz and K. Rebensburg, Using only Proportional Jitter Scheduling at the boundary of a Differentiated Service Network: simple and efficient, In Proc. of 2nd European Conf. on Universal Multi service Networks (ECUMN), pp.116-123, Colmar, France, April 2002.

[16] Y. Moret and S. Fdida, A proportional queue control mechanism to provide differentiated services, In *Proceedings of the International Symposium on Computer and Information Systems (ISCIS)*, pages 17–24, Belek, Turkey, October 1998.

[17] T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Barg havan, Delay differentiation and adaptation in core stateless networks, In Proceedings of IEEE INFOCOM 2000, pp. 421–430, Tel-Aviv, Israel, April 2000.

[18] H. T. Ngin and C. K. Tham, Achieving proportional delay differentiation efficiency, Proceedings of the 10th IEEE International Conference on Networks (Icon 2002), pp. 169-174, Singapore, August, 2002.

[19] C. Dovrolis, Proportional differentiated services for the Internet, PhD thesis, University of Wisconsin-Madison, December 2000.

[20] U. Bodin, A. Jonsson, and O. Schelen, On creating proportional loss differentiation: predictability and performance, In Proceedings of IWQoS 2001, pp. 372–386, Karlsruhe, Germany, June 2001.

[21] Kumar, J. Kaur and H. Vin, End-to-end Proportional Loss Differentiation, Technical Report. TR-01-33, University of Texas, February 2001.

[22] J. Liebeherr and N. Christin, "JoBS: Joint buffer management and scheduling for differentiated services", In Proceedings of IWQoS 2001, pp. 404–418, Karlsruhe, Germany, June 2001.

[23] L.X. Wang, A Course in Fuzzy Systems and Control, Prentice-Hall, Englewood Cli.s, NJ, 1997.

[24] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neurofuzzy and Soft Computing* (Prentice Hall: 1997)

[25] M.H.Yagmaee, M.Baradran," Fuzzy Jobs: A Fuzzy Extension to the Jobs Algorithm", in IAENG International Journal of Computer Science (IJCS), Vol 34, No.1, August 2007

[26] NS-2 Network simulator http://www.isi.edu/nsnam/ns/