Fuzzy Jobs: A Fuzzy Extension to the Jobs Algorithm

Mohammad Hossein Yaghmaee , Ghazale Khojasteh Tousi

Abstract— The relative differentiated service model 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. Relative service differentiation is a simple and easily deployed approach compared to the absolute differentiation service. One of the famous existing algorithms for both buffer management and scheduling in the relational differentiated service is Jobs algorithm. The Jobs can support both relative and absolute constraints. In this paper by using the capabilities of fuzzy logic systems, 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. The proposed fuzzy system can support both relative and absolute constraints. Simulation results confirm that the proposed fuzzy system, can provide better delay differentiated than the Jobs algorithm.

***Index Terms*— Quality of Service (QoS), Proportional Differentiated Services, Scheduling Mechanisms, Buffer Management, Fuzzy system**

I. Introduction

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 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 [1].

In the last few years, the growth of the Internet and the use of new services such as e-business, voice over IP (VoIP)[2] and multimedia applications has risen the need to support QoS requirements and to accommodate different service levels. 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:

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

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

Mohammad Hossien Yaghmaee is the associate professor at Computer Department of Ferdowsi University of Mashad, IRAN, (e-mail: hyaghmae@ferdowsi.um.ac.ir). This research was in part supported by a grant from Institute for Studies in Theoretical Physics and Mathematics (I.P.M)

Ghazale Khojasteh is with the Computer Department of Azad Islamic University of Mashad, IRAN. e-mail: ghazale_khojaste@yahoo.com

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].

In the case of proportional delay differentiation [11], defined $\bar{q}_i(t,t+\tau)=1/\bar{d}_i(t,t+\tau)$ where $\bar{d}_i(t,t+\tau)$ is the average queuing delay of the class i packets that departed in the time interval $(t,t+\tau)$. If there are no such packets, $\bar{d}_i(t,t+\tau)$ is not defined. The proportional delay differentiation model states that for all pairs of classes and for all time intervals (t, t + τ) in which both $\bar{d}_i(t,t+\tau)$ and $\bar{d}_j(t,t+\tau)$ are defined, the following equation is satisfied:

$$\frac{\bar{d}_i(t,t+\tau)}{\bar{d}_j(t,t+\tau)}=\frac{\delta_i}{\delta_j} \tag{2}$$

where the parameters $\{\delta_i\}$ are the Delay Differentiation Parameters (DDPs), being ordered as $\delta_1 > \delta_2 > \ldots > \delta_N$.

In the case of proportional loss rate differentiation [12], defined $\bar{q}_i(t,t+\tau)=1/\bar{l}_i(t,t+\tau)$ where $\bar{l}_i(t,t+\tau)$ is the fraction of class i packets that were backlogged at time t or arrived during the interval (t, t + τ), and were dropped in this time interval. In this case, the proportional loss rate differentiation takes the form:

$$\frac{\bar{l}_i(t,t+\tau)}{\bar{l}_j(t,t+\tau)}=\frac{\sigma_i}{\sigma_j} \tag{3}$$

where the parameters $\{\sigma_i\}$ are the Loss rate Differentiation Parameters (LDPs), being ordered as $\sigma_1 > \sigma_2 > \ldots > \sigma_N$. In the case of proportional jitter differentiation [14], defined $\bar{q}_i(t,t+\tau)=1/\bar{j}_i(t,t+\tau)$ where $\bar{j}_i(t,t+\tau)$ is the average jitter of the class i packets that departed in the time interval $(t,t+\tau)$. In this case, the proportional jitter differentiation takes the form:

$$\frac{\bar{j}_i(t,t+\tau)}{\bar{j}_j(t,t+\tau)}=\frac{\Delta_i}{\Delta_j} \tag{4}$$

where the parameters $\{\Delta_i\}$ are the jitter differentiation parameters (JDPs), being ordered as $\Delta_1 > \Delta_2 > \ldots > \Delta_N$.

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.

This paper, proposes a new algorithm for scheduling and buffer management in IP Diffserv network. 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 algorithm scheduling packets are based on jitter. The proposed method that based on Jobs, 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 JDP and LDP , these classes can be separated based on delay as well.

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.

The reminder of this paper is organized as follow. In section two, at first we explain the property of fuzzy systems and then we introduce the proposed fuzzy implementation of Jobs algorithm. In section 3, by using computer simulation, we compare the performance of both algorithms. Finally section 4, concludes the paper.

II. The Proposed Fuzzy Jobs

In this section we introduce the proposed fuzzy implementation of Jobs algorithm. The proposed algorithm is called Fuzzy Jobs. The main objective of the proposed Fuzzy Jobs is to enhance the delay differentiation of the Jobs algorithm. As the proposed Fuzzy Jobs, uses the fuzzy controller, we first explain the properties of fuzzy controllers. The structure of a typical fuzzy controller is shown in figure1. As shown in this figure, 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:

$$f(X) = \frac{\sum_{l=1}^{M} \bar{y}^l (\min_{j=1}^{n} \mu_{A_j^l}(x_j))}{\sum_{l=1}^{M} (\min_{j=1}^{n} \mu_{A_j^l}(x_j))} \quad (5)$$

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 $(l = 1,2,..,M)$.

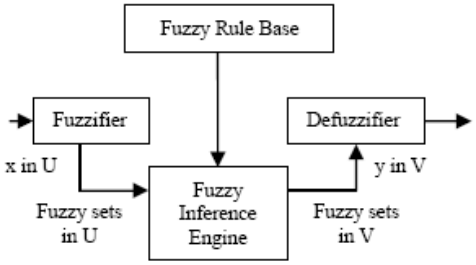


Figure1. Structure of a fuzzy controller

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:

$$e_1[i] = (Rout\_th[i] - Rout[i])/1000000 \quad (6)$$

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 normalized by dividing to 1000000. 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:

$$e_2[i] = (delay[i])/ADC[i] \quad (7)$$

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. For relative traffic classes, $e_2[i]$ is defined as below:

$$e_2[i] = (delay[i]/delay[i-1]) - RDC[i] \quad (8)$$

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 structure of the proposed fuzzy system is shown in figure 2.

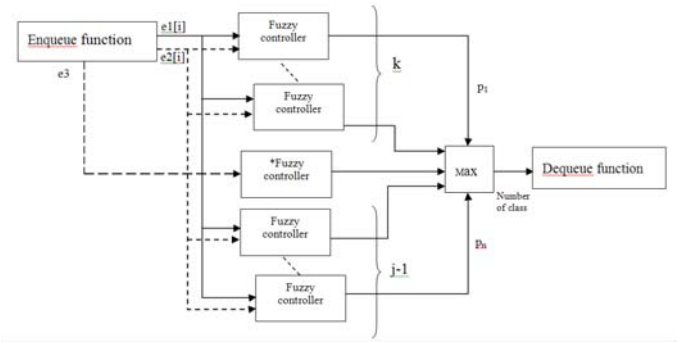
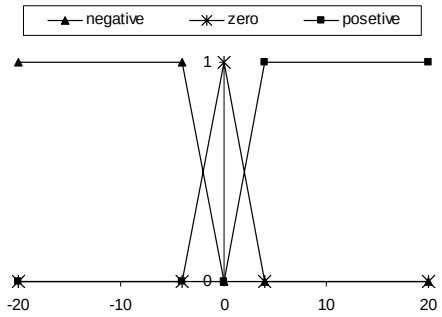


Figure 2. The structure of the proposed fuzzy system

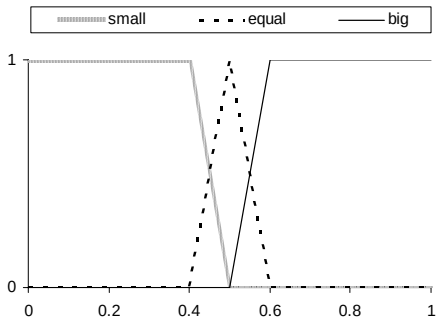
In figure 2, it is assumed that K classes use ADC and J classes use RDC. The fuzzy controller which has been specified with a star, is assigned to the first class that relational delay has been defined for it. Note that K+j=n , where n is the total number of traffic classes. Enqueue function is one of the Jobs's functions where e_1 and e_2 are calculated in this function. The biggest fuzzy controller's output is entered to the Dequeue function. This function is responsible to service the proper packet.

The membership functions of $e_1[1],...,e_1[n]$ و $e_2[1],...,e_2[n]$ and e_3 are shown in figure 3.

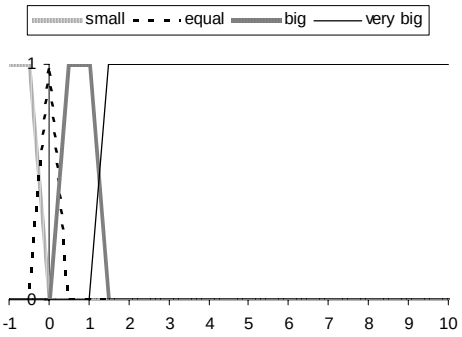
The membership function of the output signal is plotted in figure 4.



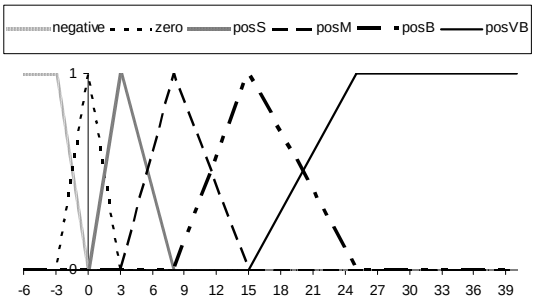
(a) Membership function for $e_1[1],...,e_1[n]$



(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

Figure 3. The membership functions of inputs e_1 , e_2 and e_3

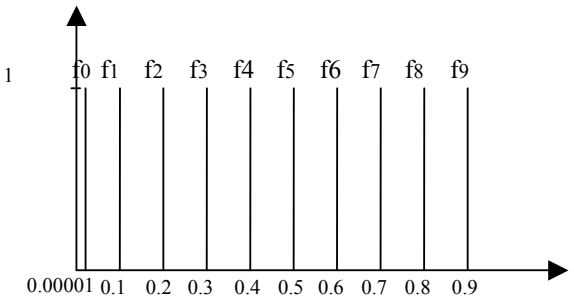


Figure 4. The output's membership function (p1,…,pn)

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

III. Simulation Results

To evaluate the performance of the proposed Fuzzy Jobs, we used the ns2[24] network simulator. The network topology used in the simulation is shown in figure 5.

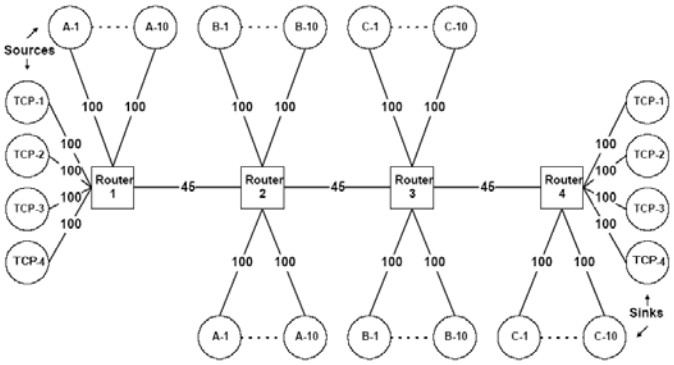


Figure 5. Network topology used in the simulation

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 1. 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.

Table1. 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	ON/OFF	1000pkts	200ms
A-2,A-3	2	TCP	ON/OFF	1000pkts	200ms
A-4,A-5, A-6	3	TCP	ON/OFF	1000pkts	200ms
A-7,A-8, A-9,A-10	4	TCP	ON/OFF	1000pkts	200ms

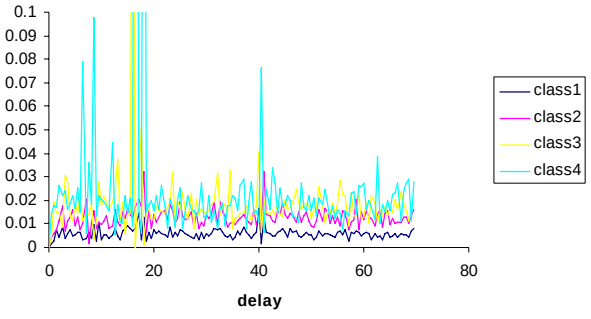
Different experiments were performed. In all experiments we used the same network topology and traffic parameters given in figure5 and table1.

A. *Experiment 1*

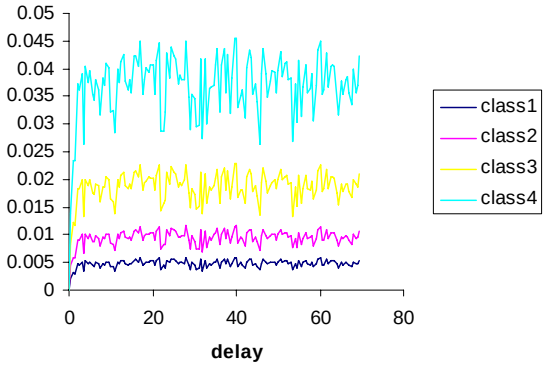
In the first scenario we only consider the RDC. The service guarantees of traffic classes are as below:

Class-4 Delay ≈ 2 × Class-3 Delay
Class-3 Delay ≈ 2 × Class-2 Delay
Class-2 Delay ≈ 2 × Class-1 Delay

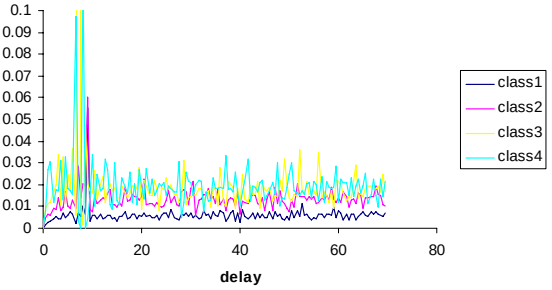
In figure 6, for all routers and for both Jobs and Fuzzy Jobs, the delay of classes is given. As the input traffic load to the Router 4 is less that its output link capacity, so there is not any queuing in this router and the delay is zero.



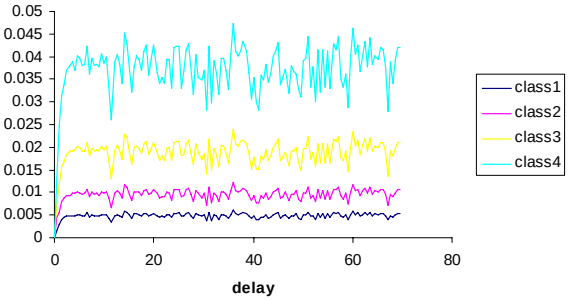
(a) Router1- Delay of classes (Jobs)



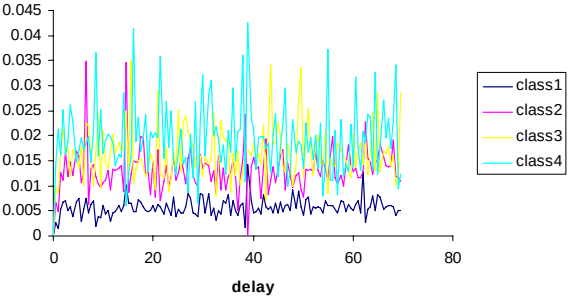
(b) Router1- Delay of classes (Fuzzy Jobs)



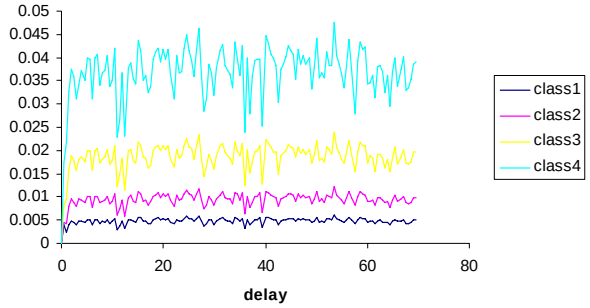
(c) Router2- Delay of classes (Jobs)



(d) Router2- Delay of classes (Fuzzy Jobs)



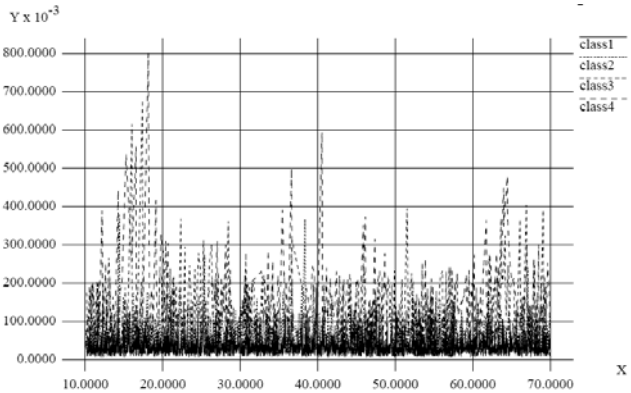
(e) Router3- Delay of classes (Jobs)



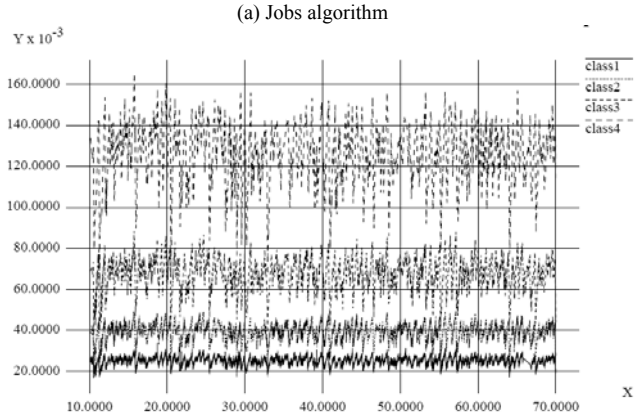
(f) Router3- Delay of classes (Fuzzy Jobs)

Figure 6. Delay of classes for both Jobs and Fuzzy Jobs algorithms (Experiment1)

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 7 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) Jobs algorithm



(b) Fuzzy Jobs algorithm

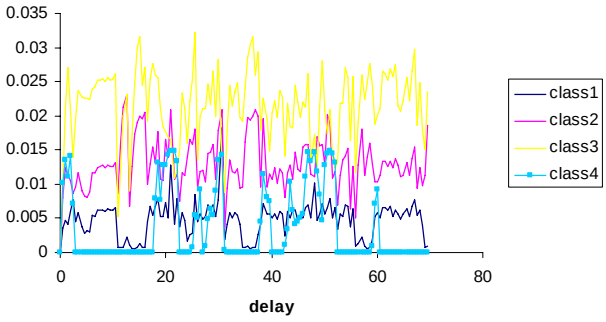
Figure 7. The end-to-end delay of flows (Experiment 1)

B. *Second Experiment*

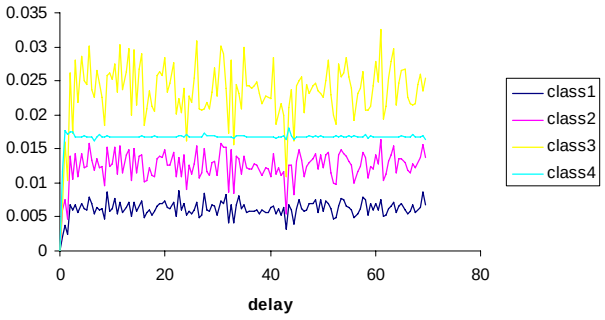
In the second scenario, we consider both ADC and RDC. In the first part of this experiment, we consider the following class of service constraints:

$$\text{Class-3 Delay} \approx 2 \times \text{Class-2 Delay}$$
$$\text{Class-2 Delay} \approx 2 \times \text{Class-1 Delay}$$
$$\text{Class-4 Delay} \leq 30 \text{ ms}$$

In this case for traffic classes 1,2,3 the RDC is defined while for traffic class 4 only the ADC is defined. In figure 8, for Router 1, and for both algorithms the delay of classes are plotted versus simulation time. As it can be seen in this figure, for the proposed Fuzzy Jobs, the delay of class 4 is less than predefined value. Furthermore it can be seen that the proposed algorithm has better fairness. This means that for the proposed algorithm the delay of all classes are close to the predefined values.



(a) Delay of classes for Jobs algorithm



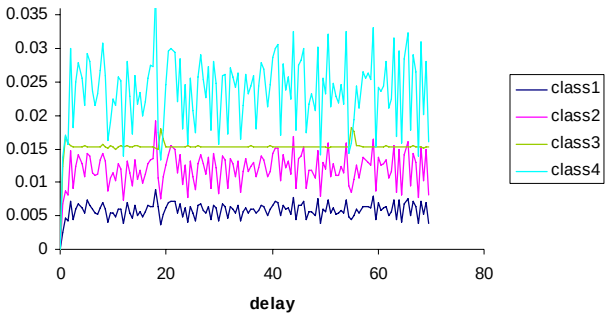
(b) Delay of classes for Fuzzy Jobs

Figure 8. Delay of classes for Jobs and Fuzzy Jobs (Experiment 2)

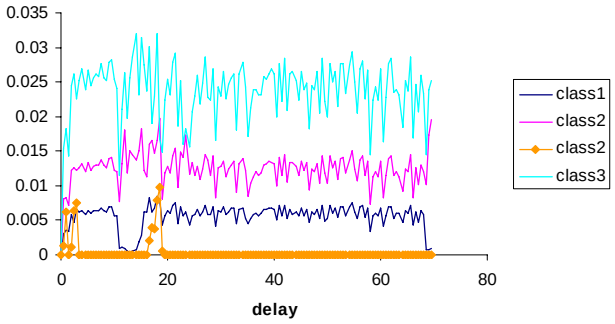
In the second part of second scenario, we consider the following service classes:

$$\text{Class-4 Delay} \approx 2 \times \text{Class-2 Delay}$$
$$\text{Class-2 Delay} \approx 2 \times \text{Class-1 Delay}$$
$$\text{Class-3 Delay} \leq 20 \text{ ms}$$

In this experiment, for traffic classes 1,2,4 the RDC are defined while for traffic class 3 only the ADC is defined. In Figure 9, for both algorithms the delay of classes in Router1 is plotted versus simulation time.



(a) Delay of classes for Jobs algorithm



(b) Delay of classes for Fuzzy Jobs

Figure9. Delay of classes for Jobs and Fuzzy Jobs (Experiment 2)

Similar to previous experiment, it is clear that the proposed Fuzzy Jobs has better performance than the traditional Jobs.

IV. CONCLUSION

Joint of Buffer Management and Scheduling (Jobs) is a powerful algorithm that manages dropping and scheduling in one step. The Jobs algorithm assumes per-class buffering of arriving traffic and serves traffic from the same class in a First-Come-First-Served order. It allocates to each traffic class a guaranteed service rate. The service rate guarantees are adjusted over time and may be changed as often as after each traffic arrival. Within the context of Jobs, there is no admission control and no policing of traffic. The set of relative or absolute performance requirements are given to the Jobs algorithms as a set of per-class QoS constraints. The set of constraints given to Jobs can be any mix of relative and absolute constraints. In this paper 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.

V. REFERENCES

[1] 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.

[2] 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.

[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/drafti2-qbone-arch-1.0.html, Aug. 1999

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

Mohammad Hossein Yaghmaee was born on July 1971 in Mashad, Iran. He received his B.S. degree in Communication Engineering from Sharif University of Technology, Tehran, Iran in 1993, and M.S. degree in communication engineering from Tehran Polytechnic (Amirkabir) University of Technology in 1995. He received his PhD degree in communication engineering from Tehran Polytechnic (Amirkabir) University of Technology in 2000. He has been a computer network engineer with several networking projects in Iran Telecommunication Research Center (ITRC) since 1992. November 1998 to July1999, he was with Network Technology Group (NTG), C&C Media research labs., NEC Corporation, Tokyo, Japan, as visiting research scholar. He is author of two books. Furthermore, he has published more than 55 technical papers. His research interests are in traffic and congestion control, high speed networks including ATM and MPLS, Quality of Services (QoS) and fuzzy logic control.

Ghazale Khojasteh Tousi was born in Mashad, Iran. She received her B.S. degree in Computer Engineering from Azad University of Mashad and her M.S. degree from Azad University of Mashad in 2007.