Quality of Service in a DiffServ Domain using Policy-Based Management

Marcial P. Fernandez[a], Aloysio de C.P. Pedroza[a b] and José F. Rezende[a]
{marcial, aloysio, rezende}@gta.ufrj.br

[a]Grupo de Teleinformática e Automação (GTA)
Universidade Federal do Rio de Janeiro (UFRJ)
COPPE/PEE - Programa de Engenharia Elétrica
C.P. 68504 - CEP 21945-970 - Rio de Janeiro - RJ - Brasil
Tel: +55 21 260-5010 Fax: +55 21 290-6626

[b]Departamento de Eletrônica /EE-UFRJ

The Differentiated Services architecture has been proposed to offer quality of service in the Internet. Most works on Diffserv (DS) handles QoS guarantees in a per node basis, which assumes that assuring QoS in a single node also leads to the desired QoS in the entire DS domain. Nevertheless, this is not always true. This paper proposes a framework that offers QoS in a DS domain using Policy-based Management and fuzzy logic techniques. The QoS controller reconfigures all DS nodes according to ingress traffic and domain policies. Policy Based Management is used in this framework to provide QoS in DS domain, controlling heterogeneous equipments of different manufacturers. The performance and functionalities of a prototype are shown by simulation of a voice over IP application.

Keywords: Quality of Service, Routing and Congestion Control, Network Management

1. INTRODUCTION

The current Internet architecture does not offer the quality of service (QoS) required by multimedia applications. The Differentiated Services (DiffServ)[1] is a proposal that aims at providing different service levels to flow aggregations. The DiffServ architecture, however, needs consistent resource provisioning in order to offer QoS in a whole domain or across inter-connected domains. Thus, to increase the resource utilization while avoiding violation of the service quality, traffic conditioners at the edges and mechanisms in the core of the network must be frequently reconfigured. With the equipment heterogeneity and the complex network topologies, the reconfiguration process can be a complex task, and it can lead to severe instabilities and even more QoS violations. One possible solution is to use management actions on the entire domain to exert QoS control. Policy-based management has been proposed as the infrastructure to guarantee QoS in a DiffServ and IntServ architecture.

This work proposes to dynamically adjust nodes settings inside a DiffServ domain. A fuzzy controller, which configures all the domain nodes dynamically using policy-based management architecture, is introduced. The fuzzy logic is used due to the uncertainty and inaccuracy characteristics of the data flow estimates. Fuzzy logic control is an interesting approach to adjust variables of controllers. Compared to a conventional digital controller, the fuzzy controller

presents advantages in handling inaccurate variables, while keeping low the complexity of the system. Our fuzzy controller is tested by simulation. The validation is achieved using a real situation, where delay, jitter and loss rate of voice flows competing with other non-sensitive delay traffics are evaluated.

The fuzzy controller is specified by the definition of fuzzy sets (membership function) and a set of rules according to a defined policy. The change in those configuration parameters makes it possible to adjust the behavior of the controller in order to make the action more or less aggressive. This behavior is defined by administrative decisions in a policy-based management framework. The use of a fuzzy controller to guarantee QoS in a network was introduced by Vasilakos and Anagnostakis[2]. However, it was applied to a Traffic Engineering environment, which does not consider a policy-based management framework.

This work is organized as follows: section 2 presents a summary on Policy-Based Management and existing problems to control QoS in the Internet; section 3 shows the QoS fuzzy controller architecture; section 4 shows the simulation model and the proposed fuzzy controller. Section 5 shows the results of simulation and, finally, section 6 presents the conclusions and suggestion for future works.

2. QoS CONTROL IN THE INTERNET

This section shows the policy-based management architecture and works related to QoS control in the Internet using fuzzy approach.

2.1. Policy-based management

The policy-based management technique[3] describes a methodology to specify behavior parameters in an abstract way. In a heterogeneous system, built with several equipments from different manufacturers, those policies can result on different configuration settings according to the characteristics of each managed device. As the policy framework consider that messages would be changed in times of weeks or months the scalability is maintained.

One application suggested for Policy-Based Management is the QoS control on IP networks. Rajan et al.[4] give an overview of Policy-Based Management used to control QoS on DiffServ and IntServ environments. Blight and Hamada [5] study the Policy-Based framework scalability in public networks.

Policy-based management system architecture was proposed by the IETF[6]. The PDP (Policy Decision Point) is the key element of this architecture which executes most of the controlling tasks in the system. This work proposes a fuzzy controller that has the role of a PDP in a policy-based management architecture.

2.2. QoS Control in the Internet and Related Works

Controlling QoS parameters is a complex task, because it requires ingress traffic estimation during a given period. Due to the traffic randomness, the measurement tend to be unreliable.

Guérin and Orda [7] showed the effects of inaccuracy and uncertainty of the traffic on network QoS. This work shows that considering only bandwidth requirements, the inaccuracy does not influence the results. However, when a data flow requires end-to-end delay guarantees, the path selection process will become intractable by the inaccuracy of traffic. Lorenz and Orda [8] also showed that the uncertainty of network state makes difficult the resource provisioning to guarantee end-to-end delay in a domain. However, they showed that decomposing the delay requirements into local restrictions and defining a probability distribution is possible to

establish an exact solution. The algorithm, however, is too complex to be processed in a node, leading to the use of a polynomial approximation, which is proposed.

The uncertainty and inaccuracy problem led us to propose a solution based on fuzzy logic. Several works have presented controllers based on fuzzy logic as for example Li and Nahrstedt [9, 10]. The authors, however, focus on the configuration environment and they did not deal with the control of network resources. Cheng and Chang [11] showed a fuzzy controller to configure the parameters in an ATM network. Their work, in spite of dealing with all the network parameters, supposes the use of an ATM network, which already offers QoS. Vasilakos and Anagnostakis [2] introduced a fuzzy controller to define the best path to obtain QoS guarantees. This proposal, used in Traffic Engineering environment, applied a genetic algorithm to the traffic history to define the fuzzy controller parameters.

Our work proposes a Fuzzy controller suitable for DiffServ and Policy-Based Management framework. The results shows an improvement in end-to-end QoS metrics to a time sensitive traffic crossing a DiffServ domain.

3. QoS CONTROLLER ARCHITECTURE

In this section, we show the QoS parameters measures in a DiffServ node for the node types in the architecture - the edge and the core nodes. The controller architecture, in spite of being specified for DiffServ, may be adapted for any mechanism that uses class based queues.

3.1. Controlling a DiffServ node

The controller in the DiffServ architecture is shown in Figure 1. In the DiffServ architecture, all nodes have a different queue for each class; a classifier forwards the packets into the respective queue and a scheduler selects packets from the queues. Besides the previous functions, the edge nodes contain a marker, that marks or remarks each packet, and a conditioner that keeps the input flow as contracted. The proposed architecture implements two controllers: one to control queues and the scheduler, used in the core and edge nodes, and other to control the conditioner, used to adjust the ingress traffic on edge nodes.

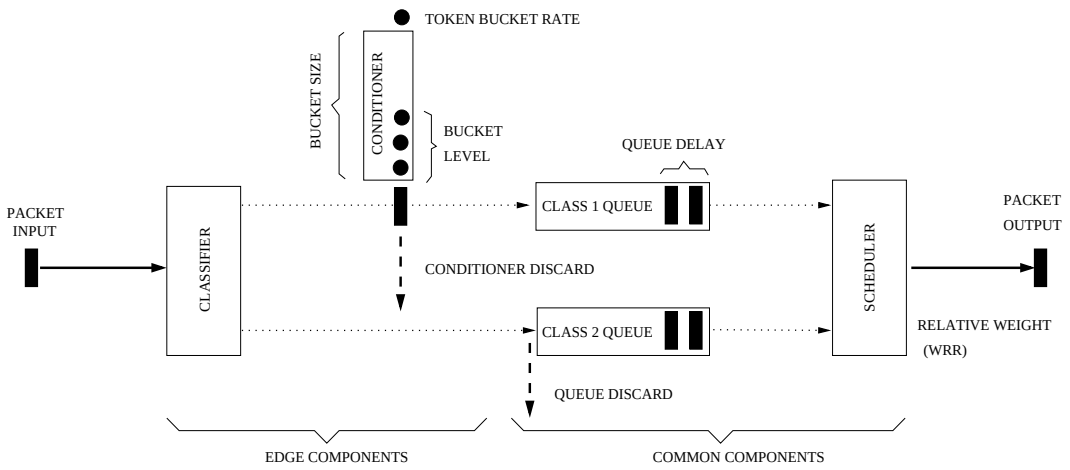


Figure 1. DiffServ node architecture

3.1.1. Scheduler Controller

The variable that controls the scheduler depends on the type of mechanism used. The scheduler in our experiment should be WRR (Weighted Round Robin) or WFQ (Weighted Fair-Queuing) type, because we can change the queue weights. The queues are served according to a configured weight that can be changed during operation. The packet delay and discard for each queue (class) can be controlled by this weight.

The first input variable is the packet delay in the queue; other delays related to packet processing are ignored. In the same way, we could use the queue occupation, because it is directly proportional to the delay. The second input variable is the loss rate due to queue overflow.

3.1.2. Conditioner Controller

The objective of this controller is to police the input data flows entering into the domain. The control variable depends on the conditioner type, nevertheless, most of then uses a token bucket based mechanism.

The first input variable is the bucket occupation, that is, the number of tokens stored in the bucket. If the bucket is full, it means that the incoming traffic in the node is smaller than the bucket rate; if the bucket is empty, it means that the bucket rate is similar or smaller to the incoming traffic. The second variable is the number of packets discarded in the conditioner. When a packet finds an empty bucket, it is discarded. In this case we can conclude that the incoming traffic is larger than the bucket rate.

The conditioner, however, cannot set the value of the bucket rate because its purpose is to maintain the traffic as contracted. A DiffServ domain should only discard packets on the edge nodes. When there are no more resources in the domain core, we should indicate a reduction in the input rate in edge nodes, through the reduction of the bucket rate. When there is some loss into the core, the edge nodes will reduce the input rate, based on the actual input average rate. Reducing the nominal contracted rate produces no effect if the actual rate is smaller. However we could use another policy, for instance, the conditioner could maintain the rate, refuse new connections and close some of them. The policy should be defined according to a customer agreement.

4. SIMULATION MODEL

The simulation model tested the EF (Expedited Forwarding) [12] and BE (Best Effort) classes. EF is the suitable class for real-time applications, like voice over IP, as it offers small delays and delay variations in each node. The BE class represents an IP network (Internet) without any guarantee, used to compete with the EF class traffic. Lorenz and Orda [8] demonstrated that uncertainty offer constraints to QoS (delay and jitter). Then, this simulation scheme is a realistic model to test the proposed fuzzy controller.

4.1. Simulation Environment

The proposed model was tested on simulation. The platform used was the Network Simulator (NS), version 2.1b6a [13]. As the NS standard distribution does not include DiffServ resources an additional module proposed by Pieda and Ethridge[14] was enclosed. New functions were added to this model in order to provide all the needed resources.

The fuzzy library used in this work was developed with the Unfuzzy tool of Duarte[15]. This tool offers a graphic interface for prototype development (specification of the membership

functions, inference rules and the defuzzificator), besides allowing initial verification of the model. The C or C++ code generated is compiled and linked to our model.

4.2. Membership functions and Fuzzification

In our example, we defined a policy that gives maximum priority to the EF class, the priority of BE class will be reduced whenever there is reduction of EF class quality. In spite of that rule, the bandwidth for BE class should never be less than 10% of the total bandwidth. Many other policies could be defined only changing membership functions and rule base.

The proposed controller uses triangular and trapezoid fuzzy sets because they are implemented with more efficient code. We also made experiences with a Gaussian function, but the results did not justify the complexity added.

4.2.1. Fuzzy Scheduler Controller

The first input membership function, EF/BE relative weight, is shown in Figure 2(a). It represents the WRR scheduler weight of the EF queue in regard to the total weight (EF+BE). The fuzzy sets are: "VLP", very low priority; "LP", low priority; "MP", medium priority; "HP", high priority and "VHP", very high priority. As the relative weight is always a number between zero and one, normalization is not needed.

The second input membership function, delay in EF queue, is shown in Figure 2(b). This sensor reads the delay of a packet through the node. The fuzzy sets are: "LD", low delay; "MD", medium delay and "HD", high delay. The delay in EF class variable must be normalized before entering into the controller.

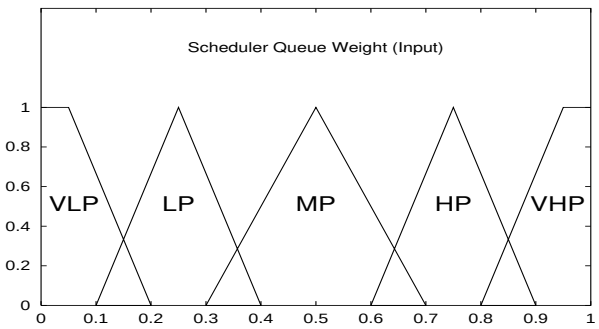
The third input membership function, discard in BE queue, is shown in Figure 2(c). This sensor reads the number of discarded packets in BE class during a time interval. The BE discard variable is normalized in regard to the maximum loss rate. The fuzzy sets are: "LD", low discard; "MD", medium discard and "HD", high discard.

4.2.2. Fuzzy Conditioner Controller

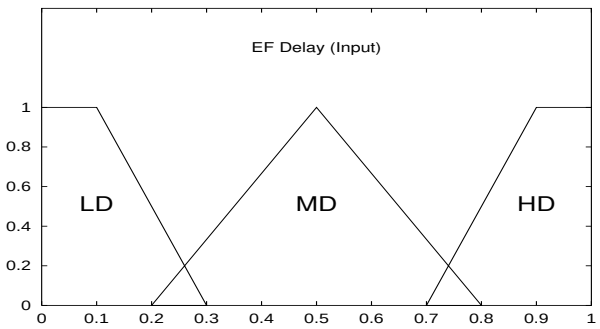
The membership function of the conditioner is similar to the scheduler function. The first input variable is the token bucket rate that reads the current token bucket rate from the conditioner. The second input is the bucket occupation that reads the level of bucket of EF class, which is a component to the measure of the real rate of incoming traffic. When the bucket level is high, it means that the rate of incoming packets is smaller than the bucket rate. The third input is the discard rate in EF queue that measures the number of discarded packets from EF class in the conditioner. When a packet is discarded, the token bucket rate is smaller than the data input rate.

4.2.3. Output Membership functions and Defuzzificator

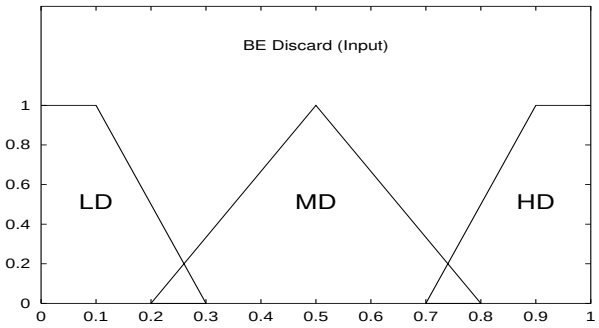
The output membership functions were also defined as trapezoid functions, by the same previous reasons. We used the center of gravity defuzzification method because it offers a result suitable for our application and it demands less floating point calculations [16]. The first output membership function, queue weight EF/BE, is showed in the Figure 2(d). It gives the weight of the WRR scheduler. The fuzzy sets are: "VLP", very low priority; "LP", low priority; "MP", medium priority; "HP", high priority and "VHP", very high priority.



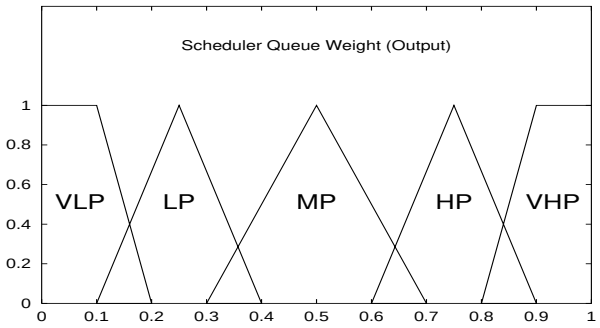
(a) Input: Queue Weight



(b) Input: EF Delay



(c) Input: BE Discard



(d) Output: Queue Weight

Figure 2. Scheduler Membership functions

The second output membership function, token bucket rate, gives the real EF class average input rate entering a node. This is the value used as reference to reduce the conditioner input rate (when there are drops in core nodes).

4.3. Rule base and Inference

Rule base is an IF-THEN rule group with fuzzy sets that represents the desired behavior of a fuzzy system. It can be defined in agreement with the administrative policy. For our example, we used a rule that gave priority to EF class in any situation, but leaving for the BE class at least 10% of the bandwidth.

The fuzzy operators were selected experimentally. We chose the maximum operator for union and intersection. Implication used the product operator and the OR and AND used the minimum operator. We have tested several other operators, however this combination offered the best result, that is, the reaction to the variations was fast and the oscillation was not excessive.

4.3.1. Fuzzy Scheduler Controller

The scheduler controller was defined with 45 rules, whose synthesis is presented as follows:

1. If the delay in EF queue is medium (MD), then the queue weight priority is increased by 1; for instance, if the priority was low (LP) it goes to medium priority (MP). And if EF delay is high (HD) the queue weight priority is increased by 2.

2. If the delay in EF queue is low (LD) and the discard rate in BE queue is medium (MD), then queue priority is reduced by 1; for instance, if the priority was medium (MP) it goes to low priority (LP). And if BE discard is high (HD) the queue priority is reduced by 2.

4.3.2. Fuzzy Conditioner Controller

The scheduler controller was defined with 27 rules, whose synthesis is presented as follows:

1. If EF delay in core nodes is low, the conditioner bucket level is low and EF queue discard is medium, then the conditioner rate is increased by 1 and if EF queue discard is high, the conditioner rate is increased by 2.

2. If EF delay in core nodes is high, the conditioner bucket level is medium and EF queue discard is low, then the conditioner rate is reduced by 1 and if bucket level is high, the conditioner rate is reduced by 2.

4.4. Conventional controller

A conventional digital controller that executes an intuitive control was defined to validate our proposal. The results of the fuzzy controller compared to a situation without controller were clearly better. We used the same sample period as the fuzzy controller. This controller presents the following characteristics:

1. If the delay average of the last three samples surpasses a certain value, it calculates the slope of the adjusted line for those three points. This slope is applied to queue weight in the scheduler, increasing the EF class rate.

2. If the delay average of the last three samples is below a certain value, and the BE queue drop rate is high, it calculates the slope of the adjusted line for those three points (that should be negative). This slope is applied to the output queue weight in the scheduler.

4.5. Simulation Topology

The application voice over IP was implemented with CBR and exponential On-Off traffic over UDP protocol. The CBR traffic is the worst case for network QoS; on the other hand, the On-Off traffic is closer to a normal conversation. The voice traffic was classified into EF class and the competitive traffic CBR/UDP was classified into BE class. The topology evaluated is outlined on Figure 3 which shows a five node DiffServ domain, with two core nodes and three edge nodes. Although we consider TCP traffic more realistic to BE class, we used CBR to avoid that TCP congestion control disguises the controller performance, since TCP reduces its rate in case of congestion and UDP does not.

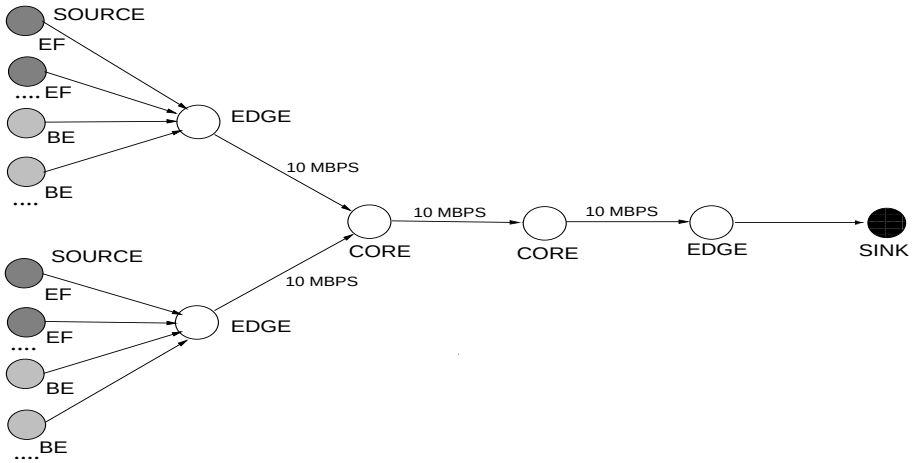
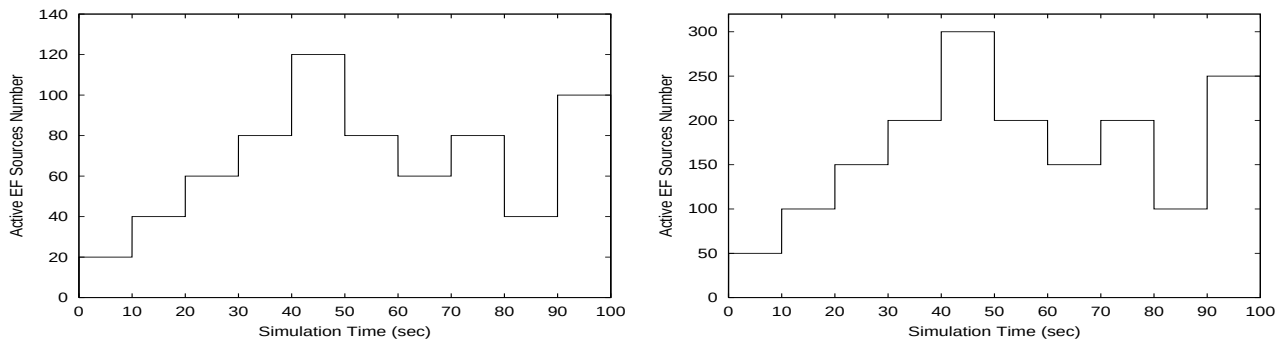


Figure 3. Simulation Topology

Figure 4 shows the number of voice traffic sources (class EF) during the simulation. The changes in traffic is used to demonstrate the controller operation in a real situation. Each CBR voice source was defined with 64 Kbps rate (PCM channel). In On-off source, we used 64 Kbps rate with burst time of 600 ms and idle time of 400 ms, giving an average rate of 25,6 Kbps. The size of the packet is 576 bytes for both cases. While the CBR traffic varied from 20 to 120 sources, the On-Off traffic varied from 50 to 300 sources, because the average rate of the On-Off traffic is approximately 2.5 times minor than CBR rate. We used 300 competitive BE sources with rate of 64 Kbps. The delay of each link of 10 Mbps is 5 ms. All queues have a maximum size of 400 packets giving a maximum delay of 23 ms for each node. The simulation model used WRR scheduler, Drop Tail queues in both classes and Token Bucket conditioner for EF class (the BE class was not conditioned).

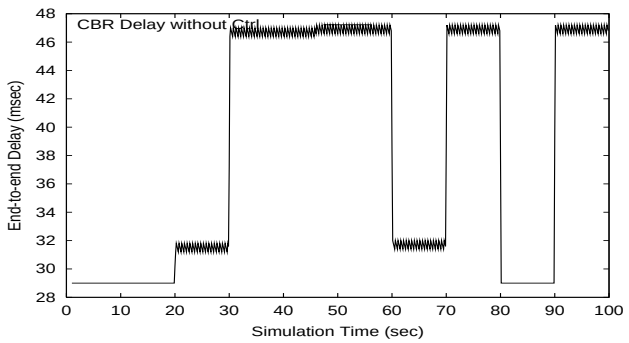


(a) CBR Traffic

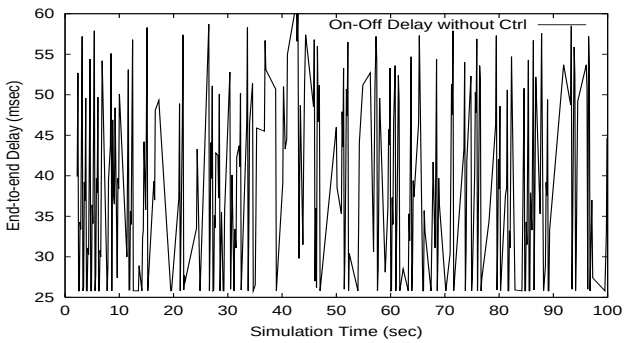
(b) On-Off traffic

Figure 4. Number of EF traffic sources during the simulation

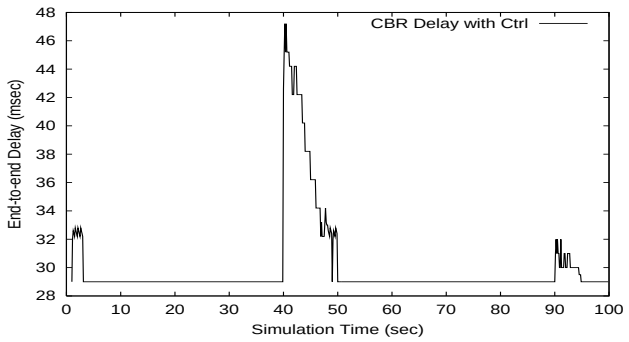
5. RESULTS



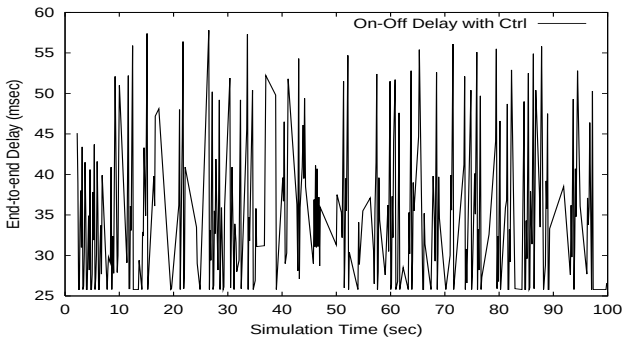
(a) CBR Traffic without controller



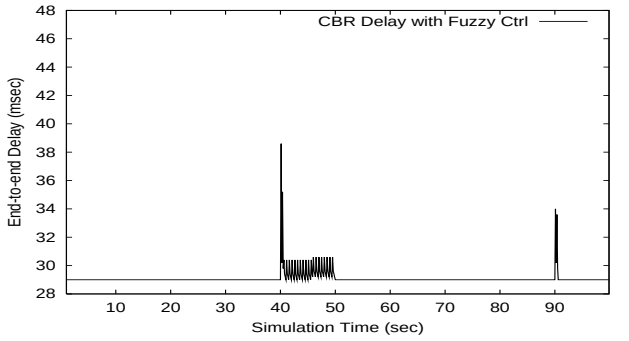
(b) On-Off traffic without controller



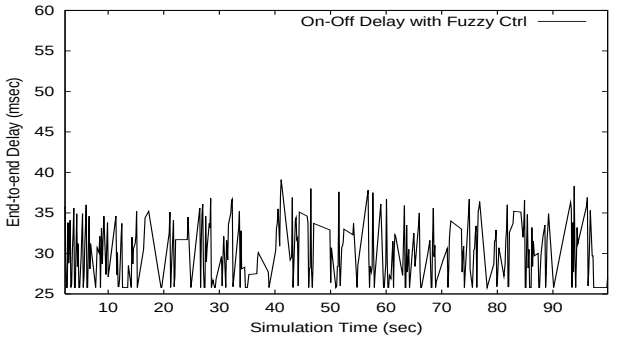
(c) CBR Traffic conv. controller



(d) On-Off Traffic conv. controller



(e) CBR Traffic fuzzy controller



(f) On-Off Traffic fuzzy controller

Figure 5. End-to-end delay of EF class

The evaluated measures were end-to-end delay and loss rate for the EF class. The measures of jitter can be deduced from delay graphs. For each evaluation, we show CBR and exponential

On-off traffic graphs. We show the graphs of a DiffServ domain without controller, with conventional controller and with the fuzzy controller. All simulations began with scheduler configuration with 50% of the bandwidth for each class. To eliminate measures with an empty network, the measures always began 5 seconds after the beginning of the simulation.

The sampling period influences the controller stability. When the period is too long or too short, it may have oscillations. The period used in this work was of 1.0 s, but we can reduce the period without penalty. We remember that this period refers to controller operation, not to the message transmission.

5.1. End-to-end delay on EF class

The graph on Figure 5(a,b) shows the end-to-end delay of voice traffic classified in EF class from source to destination. This graph shows the result without controller. Figure 5(a) shows CBR traffic and Figure 5(b) exponential On-Off traffic. We can see in Figure 5(a) the flat graph which confirms that the queue reached its maximum capacity. In On-off traffic, the delay varies due to the exponential distribution of the sources entering into the domain.

The graph on Figure 5(c,d) shows the end-to-end delay with a conventional controller. We noticed in Figure 5(c) an improvement of the delay comparing to the case without controller; the controller corrects the traffic variations, but very slowly. In On-off traffic, in Figure 5(d), the delay presents an improvement too.

The graph on Figure 5(e,f) shows the end-to-end delay with fuzzy controller. We noticed in Figure 5(e) an improvement of the delay comparing to the case without controller and with a conventional controller. The fuzzy controller corrects all variations quickly, keeping delay values closer to the propagation delay of the network (25 ms). The peak on 40 s and 90 s is due to the controller sampling period, which executes correction controls only each 1 second. In On-off traffic, Figure 5(f), the average delay is lower to the previous cases.

5.2. Discard rate on EF class

The graph on Figure 6 shows the loss rate of the voice traffic on EF class. Without controller, the loss rate with traffic CBR, Figure 6(a), occurs on every period when queues reach their limit. In On-off case, Figure 6(b), the discard only occurs when network resources become exhausted.

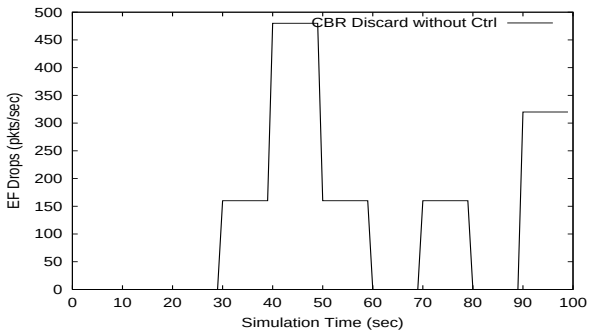
The graph on Figure 6(c,d) shows the loss rate with the conventional controller. We can notice a decrease of EF discard rate for both traffics comparing to the previous case.

The graph on Figure 6(e,f) shows the loss rate with the fuzzy controller. Considering CBR traffic, we had an improvement in discard rate, and considering the On-off traffic, the losses felt to zero.

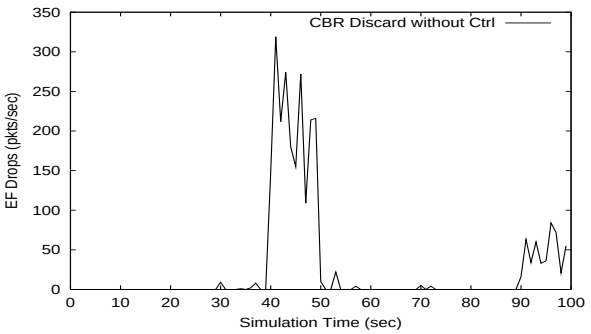
6. CONCLUSION AND FUTURE WORKS

We showed, in this paper, a fuzzy controller that reconfigures router parameters to offer better quality of service inside a DiffServ domain. The use of fuzzy logic improves the handling of inaccuracy and uncertainties of the ingress traffic in a domain. The controller kept low complexity, maintaining DiffServ scalability characteristic.

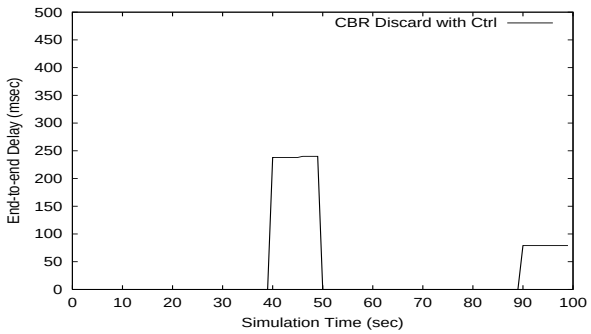
The policy-based management allow the controller to be generic, defining the domain behavior in agreement with administrative decisions. Furthermore, the behavior can be easily changed in the whole domain, during normal system operation.



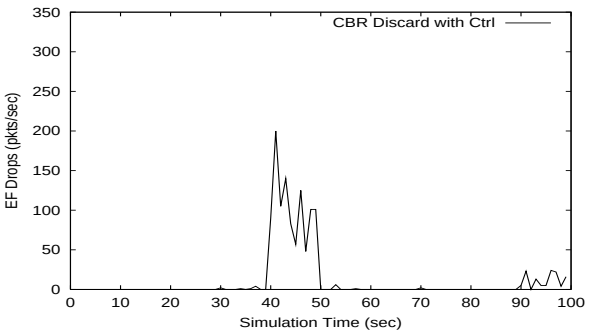
(a) CBR Traffic without controller



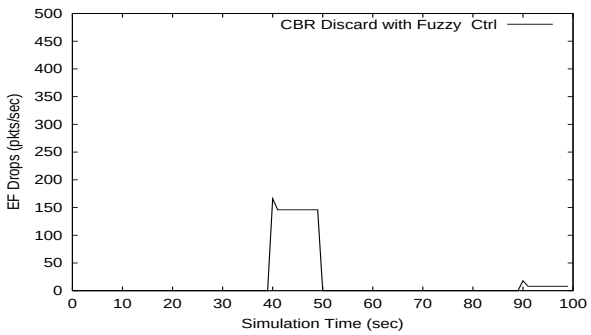
(b) On-Off Traffic without controller



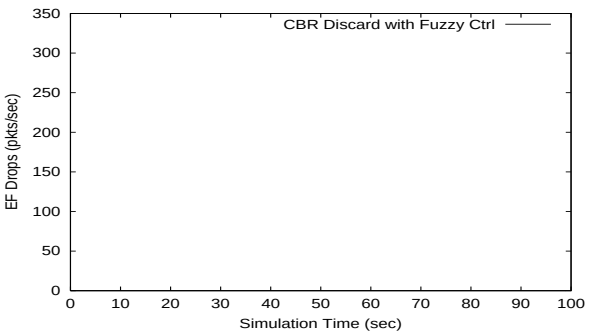
(c) CBR Traffic conv. controller



(d) On-Off Traffic conv. controller



(e) CBR Traffic fuzzy controller



(f) On-Off Traffic fuzzy controller

Figure 6. Discards in the EF class

The simulations used to validate the model considered an example with EF and BE classes. Delay and jitter measurements are affected by inaccuracy and uncertainty of ingress traffic into the domain [8]. The results obtained through simulation demonstrated the functionality of the proposal showing an improvement in QoS metrics.

The results obtained using Drop Tail queues, WRR scheduler and Token Bucket conditioner,

already reached improvement in QoS metrics. If we use WFQ scheduler, the fairness will be maintained even with variable packet length.

As future work a new controller will be defined including support to other DiffServ classes, like AF (Assured Forwarding). This class has a different philosophy, forcing the controller to deal with variables different from those considered in this paper. Thus we will have a complete controller, able to adjust all DiffServ parameters dynamically, according to the traffic changes and a given policy.

REFERENCES

1. BLAKE,S., BLACK, D., ,CARLSON, M., et al, "An Architecture for Differentiated Services", Request for Comments 2475, December 1998.
2. VASILAKOS, A., ANAGNOSTAKIS, K., "Evolutionary-Fuzzy Prediction for Strategic Inter-Domain Routing: Architecture and Mechanisms", WCCI 98. Anchorage, USA, May 4-9 1998.
3. SLOMAN, M., "Policy Driven Management for Distributed Systems", Journal of Network and Systems Management, Vol. 2, No. 4, pp. 333-360, 1994.
4. RAJAN, R., VERMA, D., KAMAT, S., at al. "A Policy Framework for Integrated and Differentiated Services in the Internet", IEEE Network Magazine. USA, pp. 36-41, September/October 1999.
5. BLIGHT, D., HAMADA, T., "Policy-Based Networking Architecture for QoS Interworking in IP Management - Scalable Architecture for Large-Scale Enterprise Public Interoperation",
6. STEVENS,M., WEISS, W., MAHON, H., at al., "Policy Framework", Internet Draft draft-ietf-policy-framework-00.txt, September 1999.
7. GUÉRIN, R., ORDA, A., "QoS-based Routing in Networks with Inacurate Information: Theory and Algorithms", IEEE Infocom 97. Kobe, Japan, 1997
8. LORENZ, D., ORDA, A., "QoS Routing in Networks with Uncertain Parameters", IEEE Infocom 98
9. LI, B., NAHRSTEDT, K., "A Control-Based Middleware Framework for Quality of Service Adaptions", IEEE Journal on Select Areas in Communication, September 1997.
10. LI, B., NAHRSTEDT, K., "Dinamic Reconfiguration for Complex Multimidia Application", IEEE International Conference on Multimedia Computing and Systems 99, Vol 1, pp 165-170, July 1999.
11. CHENG, R., CHANG, C., "Design of a Fuzzy Traffic Controler for ATM Networks", IEEE/ACM Transaction on Networking, v.4 nº 3, pp 460-469, June 1996.
12. JACOBSON, V., NICHOLS, K., PODURI, K., "An Expedited Forwarding PHB", Request for Comments 2598, June 1999.
13. Network Simulator - NS Version 2, http://www.isi.edu/nsnam/ns/
14. PIEDA, P., ETHRIDGE, J., BAINES, M., et al.,"A Network Simulator Differentiated Services Implementation - Open IP", Nortel Networks, http://www7.nortel.com:8080/CTL/
15. DUARTE, O., Unfuzzy, http://ohm.ingsala.unal.edu.co/ogduarte/
16. ROSS, T., "Fuzzy Logic with Engineering Applications", McGraw-Hill, New York, 1995.