QoS Provisioning across a DiffServ Domain using Policy-Based Management

Marcial Porto Fernandez, Aloysio de Castro P. Pedroza and José Ferreira de Rezende
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

Abstract-**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, controling heterogeneous equipments of different manufacturers. The performance and functionalities of a prototype are shown by simulation of a voice over IP application in different DS topologies.**

Keywords-**QoS, DiffServ, Policy-Based Management**

I. Introduction

The current Internet architecture does not offer the quality of service (QoS) required by multimedia applications. The Differentiated Services (DiffServ) 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 complicated task, and it can lead to severe instabilities and even more QoS violations. One possible solution is to use management action on whole 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 estimate. 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 is evaluated. These QoS metrics are evaluated in two random topologies to assure the model validation.

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 possible to adjust the behavior of controller in order to turn the action more or less aggressive. This behavior is defined by administrative decisions in policy-based management framework. The use of a fuzzy controller to guarantee QoS in a network was first introduced by Vasilakos and Anagnostakis [1]. However, it is applied to a Traffic Engineering environment, which did not consider a policy-based management framework.

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

II. Quality of service in Internet

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

A. *Policy based management*

The policy-based management technique describes a methodology to specify behavior parameters in an abstract way. In a heterogeneous system, built with several equipment from different manufacturers, those policies can result on different configuration settings according to the characteristics of each managed device. One aplication suggested is the QoS control in IP networks.

Policy-based management system architecture was proposed by IETF [2]. The PDP (Policy Decision Point) is the key element of this architecture which executes most of the controlling task in the system. This work proposes a fuzzy controller that has the role of a PDP in a policy-based management architecture.

B. *QoS Control in Internet and Related Works*

Controlling QoS parameters is a complex task, since it requests ingress traffic forecast during a period. Due to the traffic randomness, the forecasts tend to be little reliable.

Guérin and Orda [3] showed the effects of inaccuracy and uncertainty of the traffic on network QoS. This work shows that considering only badwidth requirement, the inaccuracy does not influence the results. However, when a dataflow requires end-to-end delay guarantees, the inaccuracy will turns intractable the path selection process. Lorenz and Orda [4] also showed that the uncertainty of network state may difficult the provision of 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 then to the use a polynomial approximation, proposed in their paper.

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 [5]. The authors, however, focus on the configuration environment and they did not deal with the control of network resources. Cheng and Chang [6] 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 resources inside. Vasilakos and Anagnostakis [1] introduced a fuzzy controller to define the best path to offer QoS guarantees. This proposal, applied to a Traffic Engineering environment, used a genetic algorithm applied to the traffic history to define the fuzzy controller parameters.

B.1 Controlling a DiffServ node

The controller in DiffServ architecture is shown in Figure 1. In the DiffServ architecture, all nodes have a different queue for each class, a classifier put the packets into the respective queue and a scheduler gets 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 scheduler, used in the core and edge nodes, and other to the conditioner, used to adjust the ingress traffic on edge nodes.

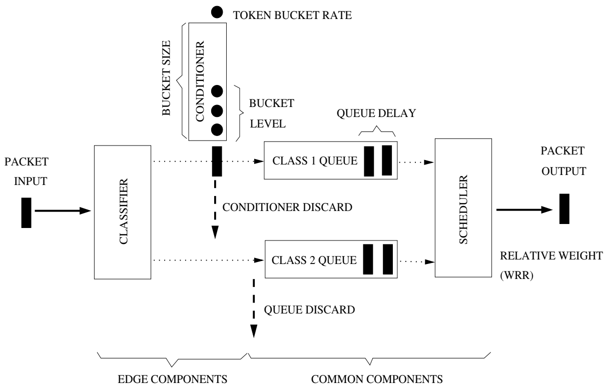


Fig. 1. DiffServ node architecture

III. Simulation Model

The simulation model tested the EF (Expedited Forwarding) and BE (Best Effort) classes. EF is the best class for real-time applications, like voice over IP, as it offers the smallest delay 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 [4] demonstrated that uncertainty offer constraints to QoS (delay and jitter). Then, this simulation scheme is a realistic model to test the proposed fuzzy controller.

A. *Simulation Environment*

The platform used was the Network Simulator (NS), version 2.1b6a [7]. As the NS standard distribution does not include DiffServ resources, it was enclosed an additional module, proposed by Pieda and Ethridge [8]. 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[9]. 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.

B. *Fuzzy Controller*

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 this rule, the bandwidth for BE class should never be less than 10% of total bandwidth. Many other policies could be defined only changing membership functions and rule base.

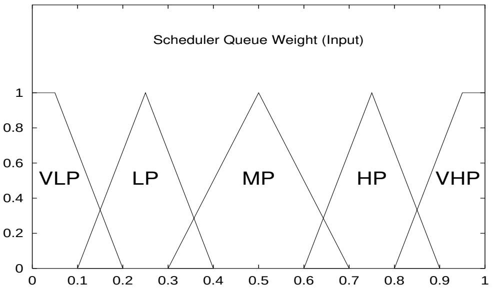
The proposed controller uses triangular and trapezoid fuzzy sets because they are implemented with more efficient code. We also made experiences with a Gaussian function, but the results did not justify the complexity added.

B.1 Scheduler Controller

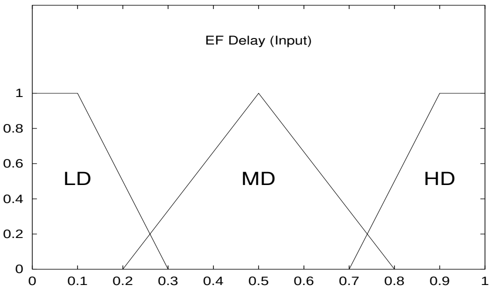
The scheduler in our experiment should be WRR (Weighted Round Robin) or WFQ (Weighted Fair-Queuing) type, because they can be controlled. The queues are served according a configured weight that can be change during operation. The packet delay and discard for each queue (class) can be controlled by this weight.

The first input variable and 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.

The second input variable is the packet delay in the queue, as other delays related to packet processing into node would be ignored. The 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 variable must be normalized.



(a) Input: Queue Weiht



(b) Input: EF Delay

Fig. 2. Scheduler Membership functions

The third input variable is the discard rate due to queue overflow. Several mechanisms of active queue management could be used, however we should consider that, if there is discards, there is shortage of resources. The membership function, discard in BE queue, reads the number of discarded packets in BE class during a time interval. The value must be normalized.

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 float point calculations than other methods. The output membership function gives the weight of the WRR scheduler.

B.2 Conditioner Controller

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 arrives the conditioner and 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 are high delay into the core, the edge nodes will reduce the input rate. We could use another policy, e.g., the conditioner could maintain the rate, refuse new connections and close some of them.

The membership function of the conditioner is similar to the scheduler function. The first input 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. 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. The last input is the maximum queue delay inside core nodes. The output membership function, token bucket rate, gives the EF class input rate entering a node.

B.3 Rule base and Inference

Rule base is an IF-THEN rule group with fuzzy sets that represent 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 selected were maximum for union and intersection, product for implication and minimum for the OR and AND function. We have tested other operators, however this combination offered the best result, that is, the reaction to the variations was fast and the oscillation was not excessive.

B.3.a 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.

B.3.b Conditioner Controller. The scheduler controller was defined with 27 rules, whose synthesis is presented as follows:

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

C. 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 controler. 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.

D. 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, also CBR/UDP, was classified into BE class. The topology evalueted was outlined on Figure 3 which shows a DS domain composed of 40 nodes, 30 core nodes and 10 edge nodes. There is five input edge nodes and 5 output edge nodes. The topologies was created with gt-itm package (bundled in NS)[7]. We used two diferents topologies whose metrics are shown in Table I.

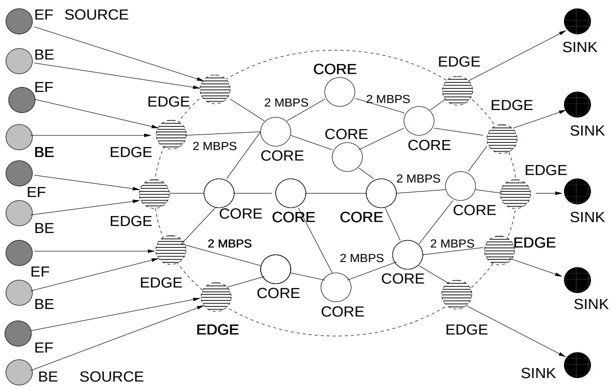


Fig. 3. Simulation Topology

TABLE I
TOPOLOGY METRICS

Topology	Type	Avg Deg	Bicomp
Topol. 1	Waxman 2	4.133	3
Topol. 2	Exponential	4.133	3

To simulate voice conections, the number of voice traffic sources (EF class) changes during the simulation time of 200 seconds. The changes in traffic showed the controller operation in a real situation. The number of voice source varies according to an exponential distribution with the following time paterns (traffic on, traffic off): (5.0,2.0), (10.0,10.0), (6.0,3.0), (20.0,20.0), (15.0,5.0) simulating telephone calling. Each CBR voice source was defined with a 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 0 to 150 sources, giving an average of 93 active sources, the On-Off traffic varied from 0 to 300 sources giving an average of 186 active sources. We used 150 and 300 competitive BE CBR sources with rate of 64 Kbps. The delay of each link of 2 Mbps is 10 ms. All queues have a maximum size of 100 packets giving a maximum delay of 225 ms in each node. The simulation model used a WRR scheduler, Drop Tail queues in both classes and Token Bucket conditioner for EF class (the BE class was not conditioned).

IV. RESULTS

The evaluated measures were the percentil of end-to-end delay and jitter of the EF class. For each evaluation, we used CBR and exponential On-off traffic. We show the tables of a DiffServ domain without controller, with a conventional controller and with the proposed fuzzy controller. All simulations began with initial scheduler configuration with 50% of the bandwidth for each class. To eliminate measures with an empty network, the measures always started 5 seconds after the beginning of the simulation.

A. End-to-end delay and Jitter on EF class

We show the simulation results by evaluating the percentil of delay and jitter for all flows. Table II and III shows the delay and jitter of CBR and On-Off traffic for topology 1. The percentil 50 means that 50% of all packets have a delay and jitter smaller than this value. The same remark is valid to percentil 90 and 95. Table IV and V shows the delay and jitter of CBR and On-Off traffic for topology 2.

TABLE II
PERCENTIL OF EF DELAY IN TOPOLOGY 1 (MSEC)

Traffic	Avg.	Per 50	Per 90	Per 95
CBR W/Ctrl	151.0	172.2	217.1	226.9
CBR Conv.	101.3	50.1	207.9	220.0
CBR Fuzzy	39.3	38.5	61.0	66.2
OO W/Ctrl	137.6	143.1	159.8	162.0
OO Conv.	127.8	140.8	159.3	161.7
OO Fuzzy	37.7	28.1	77.5	91.0

TABLE III

PERCENTIL OF EF JITTER IN TOPOLOGY 1 (MSEC)

Traffic	Avg.	Per 50	Per 90	Per 95
CBR W/Ctrl	60.0	69.7	145.1	146.3
CBR Conv.	71.3	69.7	144.5	145.7
CBR Fuzzy	1.5	0.0	2.3	2.3
OO W/Ctrl	15.3	7.8	40.2	56.8
OO Conv.	24.2	8.9	87.3	111.0
OO Fuzzy	25.7	17.7	61.7	76.9

TABLE IV

PERCENTIL OF EF DELAY IN TOPOLOGY 2 (MSEC)

Traffic	Avg.	Per 50	Per 90	Per 95
CBR W/Ctrl	173.3	205.0	300.1	353.0
CBR Conv.	139.1	167.0	263.8	274.1
CBR Fuzzy	79.6	63.9	157.2	169.3
OO W/Ctrl	176.7	167.8	273.3	289.8
OO Conv.	118.2	143.1	205.8	243.7
OO Fuzzy	59.4	51.8	113.4	132.2

B. *Discard on DiffServ Domain*

Table VI shows the loss rate of the voice traffic on EF class and default traffic on BE class in topology 1. The Table VII shows the loss rate for topology 2. We may notice a decrease of the EF discard with Fuzzy controller for both traffics comparing the experiment without and with conventional controller.

V. CONCLUSION AND FUTURE WORKS

We showed, in this paper, a study of provisioning techniques to garantee QoS in a Diffserv domain. A fuzzy controller that reconfigures router parameters was proposed to offer better QoS. 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 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 [4]. The results obtained through simulation demonstrated the functionality of the proposal showing an improvement in QoS metrics.

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.

TABLE V

PERCENTIL OF EF JITTER IN TOPOLOGY 2 (MSEC)

Traffic	Avg.	Perc 50	Perc 90	Perc 95
CBR W/Ctrl	93.3	69.7	218.3	288.5
CBR Conv.	70.5	2.3	216.0	217.7
CBR Fuzzy	42.0	2.3	143.4	144.0
OO W/Ctrl	56.6	39.0	144.7	168.9
OO Conv.	52.0	23.2	147.6	170.1
OO Fuzzy	30.9	19.4	80.3	100.3

TABLE VI

EF DROPS IN TOPOLOGY 1

Controller	CBR	On-Off
Without	53010	192405
Conventional	37611	184548
Fuzzy	14	8094

TABLE VII

EF DROPS IN TOPOLOGY 2

Controller	CBR	On-Off
Without	49859	189435
Conventional	46823	172764
Fuzzy	4579	32087

REFERENCES

[1] A. Vasilakos and K. Anagnostakis, "Evolutionary-fuzzy prediction for strategic inter-domain routing: Architecture and mechanisms," in *WCCI 98*, Anchorage, USA, May 1998.

[2] M. Stevens, W. Weiss, and M. Mamon, "Policy framework," Internet Draft draft-ietf-policy-framework-00.txt, Sept. 1999.

[3] R. Guérin and G. Orda, "Qos-based routing in networks with inacurate information: Theory and algorithms," in *IEEE Infocom 97*, Kobe, Japan.

[4] D. Lorenz and G. Orda, "Qos routing in networks with uncertain parameters," in *IEEE Infocom 98*.

[5] B. Li and K. Nahrstedt, "A control-based middleware framework for quality of service adaptions," *IEEE Journal on Selected Areas in Communications*.

[6] R. Cheng and C. Chang, "Design of a fuzzy traffic controler for atm networks," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 460–469, June 1996.

[7] NS, "Network simulator - ns version 2," http://www.isi.edu/nsnam/ns/.

[8] Peter Pieda, J. Ethridge, and M. Baines, "A network simulator differentiated services implementation - open ip," http://www7.nortel.com:8080/CTL/, Oct. 2000.

[9] O. Duarte, "Unfuzzy," http://ohm.ingsala.unal.edu.co/ogduarte/.