Optimizing Fuzzy Controllers with Genetic Algorithms for QoS Improvement

Marcial Porto Fernandez, Aloysio de Castro P. Pedroza and José Ferreira de Rezende
Universidade Federal do Rio de Janeiro, Rio de Janeiro RJ, Brasil

Abstract— **Most work on Differentiated Services (DiffServ) handles Quality of Service (QoS) provisioning on a per node basis, which assumes that this strategy would provide QoS in the whole domain. Nevertheless, this approach could fail in large domains with multiple flows aggregation and unexpected input traffic. Therefore, provisioning techniques should be used to avoid unpredicted overloads that result in QoS fluctuations. A proposal using fuzzy controllers to reconfigure DiffServ nodes according to ingress traffic and achieved QoS was presented in [1]. However, it is not easy to specify fuzzy rule bases and membership functions that optimize the controllers performance. Thus, we propose a methodology to choose optimized fuzzy controller parameters using the Wang-Mendel and genetic algorithms. Finally, we evaluate the performance of this methodology by simulation of voice over IP applications in DiffServ domains.**

Keywords—**QoS, DiffServ, Network Provisioning**

I. INTRODUCTION

THE Differentiated Services (DiffServ) [2] is a proposal that aims at providing Quality of Service (QoS) for a certain number of service classes in the Internet. This is achieved through service discrimination among flow aggregations pertaining to these classes. The specification of a limited number of classes increases substantially the scalability of this architecture. As any other proposal for QoS provisioning in the Internet, the DiffServ relies on resource allocation in network elements of the DiffServ domain (DS domain). The main difference between other existing proposals, e.g. IntServ, is that these allocations are performed on a per-class basis. In a first stage, the IP QoS model proposed by the DiffServ architecture assumes that a static allocation is to be used. However, the unpredictability and the randomness of the traffic entering in a DS domain make the dynamic resource provisioning better than a static approach.

Dynamic provisioning implies using either signaling or reconfiguring mechanisms inside the DS domain. The former approach adds complexity to the core nodes, which reduces its scalability. The latter relies on performance monitoring, which allows detecting QoS faults, to determine the need of reconfiguration. In a DiffServ domain with Policy-Based Management, one network element contains the information regarding the configuration of the different edge devices. Further, this element stores the information pertinent to the performance levels specified in the different SLAs (Service level Agreements) that are to be supported by the DS domain. Based on these information and performance monitoring, an element can control the QoS for each service class taking appropriate actions in order to minimize violations. Due to the complexity of these mechanisms, most network operators prefers resource overprovisioning. This approach, however, presents high costs, as the full capacity is not used most of the time.

Marcial Porto Fernandez, Aloysio de Castro P. Pedroza and José Ferreira de Rezende are with the Grupo de Teleinformática e Automação (GTA), COPPE/PEE - Programa de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, C.P. 68504 - CEP 21945-970 - Rio de Janeiro - RJ, Brasil. E-mails: {marcial,aloysio,rezende}@gta.ufrj.br.

In this work, we support the use of dynamic provisioning through a reconfiguration scheme, which is controlled by a Policy-Based Management system that allows for administrative decisions. In previous works [1], [3], we proposed the use of fuzzy logic controllers for the dynamic reconfiguration of edge and core routers. This reconfiguration allows for adjusting the network provisioning according to the incoming traffic and the QoS level achieved. This proposal showed good results on a simple DiffServ domain with 5 nodes [3] and on complex domains with different topologies [1].

The fuzzy logic is used due to the uncertainty associated with ingress traffic estimation and to the non linearity and lack of mathematical models able to estimate this traffic [4]. Lorenz and Orda demonstrate in [5] that this uncertainty places additional constraints on QoS provisioning. A fuzzy controller is specified by fuzzy sets definition (membership function) and a set of rules (rule base). Nevertheless, as these values are chosen by a designer, one cannot guarantee the correction and optimization of fuzzy controllers for the considered problem [4]. Thus, in order to improve fuzzy controllers performance we used a genetic algorithm to optimize them [6], [7]. To validate this optimization procedure, we evaluate packet delay, jitter and drop rate of voice-over-IP flows competing with non-sensitive delay traffic in a DS domain controlled by the proposed scheme.

This work is organized as follows: section II presents a brief description of related works on resource provisioning schemes and on the use of fuzzy logic to improve QoS; section III shows the fuzzy controller architecture and the methodology used for optimization; section IV shows the simulation model; section V shows the results of simulation; and finally, section VI presents the conclusions and suggestions for future works.

II. RELATED WORKS

Recently, several schemes concerning resource provisioning for IP networks have been proposed. The Tequila Project (Traffic Engineering for Quality of Service in Internet, at Large Scale) [8], [9], [10] aims at investigating provisioning techniques, admission control and dynamic resource management for DiffServ networks. The proposed architecture is composed of two functional groups. The first includes the SLS (Service Level Specification) Manager that is responsible for users' signature and admission control, in addition to SLS monitoring. The second functional group is responsible for resource management in long time scales (months or years). Other important functions of this group are performed by the Policy Manager. This element interprets policies defined by network operators, implements this policy in the equipments and monitors their behavior.

The RMD (Resource Management of DiffServ) framework [11], [12] supports the use of edge-to-edge signaling for resource reservation in DiffServ networks. For this purpose, this framework defines two new resource reservation protocols: PHR (Per Hop Reservation) and PDR (Per Domain Reservation) protocols. The PHR protocol, implemented in all nodes of the domain, is used to treat node attributes as an argument of PHB (Per Hop Behavior) to perform resource reservation. On the other hand, the PDR protocol manages all resource reservations at the domain level relying on the state of the reservation performed by the PHR in all domain nodes. It is implemented only on the edge nodes of the domain. The authors claim as advantages of their proposal the simplicity and the low implementation cost to assure the characteristic of good scalability.

Liao and Campbell [13], [14] propose a provisioning strategy composed by dynamic node provisioning and dynamic core provisioning algorithms. The dynamic node provisioning algorithm prevents transient violations of SLAs (Service Level Agreements) by self-adjusting per-scheduler service weight and packet dropping thresholds at core routers. The dynamic core provisioning algorithm dimensions traffic aggregates at the network ingress. This algorithm takes into account fairness issues not only among different aggregates, but also within the same aggregate whose packets take different routes in a core IP network.

Ghosh et al [15] present a survey on the use of fuzzy logic in telecommunication networks. Several works make use of controllers based on fuzzy logic. Li and Nahrstedt [16] use fuzzy logic on the configuration environment but they do not deal with network resource control. Cheng and Chang [17] uses a fuzzy controller to configure the parameters in an ATM network. Vasilakos and Anagnostakis [18] introduce a fuzzy controller to define the best path in the network that offer QoS guarantees.

III. Fuzzy QoS Controller

In this section, we present the elements in the DiffServ architecture, which are under control of fuzzy logic controllers. We also describe the two different proposed controllers. Then, we show the optimization procedure to achieve the best parametrization of these controllers.

A. *DiffServ Controller Architecture*

The controllable elements in the DiffServ architecture are shown in figure 1. In this architecture, all nodes have a separate queue for each service class; a classifier places the packets into the respective queue and the scheduler selects packets from these queues for transmission in the output links. In addition to these elements, the edge nodes contain a marker that (re)-marks each packet, and a policer that keeps the ingress traffic on edge node as contracted. The proposed architecture implements two controllers: one that controls the queues and scheduler, which is used in the core and edge nodes, and the other that controls the policer, which is only used on the edge nodes.

B. *Fuzzy Controller*

The fuzzy logic was introduced by Lofti Zadeh [19] as a generalization of the boolean logic. The difference between these logics is that fuzzy set theory provides a form to represent uncertainties, that is, it accepts conditions partially true or partially

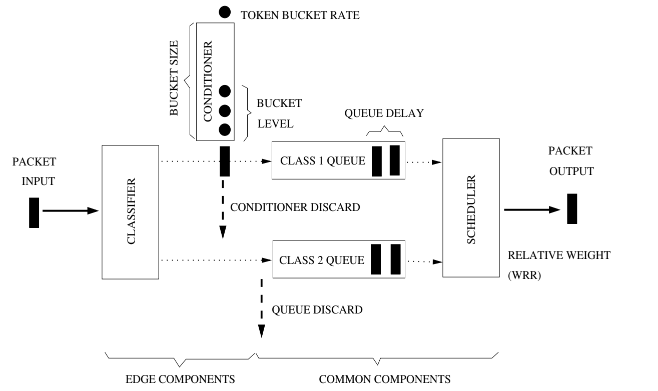

Fig. 1. Controllable elements in the DiffServ architecture

false. Fuzzy logic is the best logic to treat random uncertainty, i.e., when the prediction of a sequence of events is not possible.

In this work, we define a policy that gives maximum priority to the service class designed to carry real-time traffic, namely the EF class. The competing traffic is carried on the Best-Effort (BE) class for which no guarantees are provided. The priority of the BE class is reduced whenever there is reduction of the quality of the EF class. To avoid BE traffic starvation, the bandwidth reserved to this class should never be less than 10% of total bandwidth. Many other policies could be defined just by changing membership functions and the fuzzy rule base.

The proposed controller uses triangular and trapezoid fuzzy sets because they are implemented with more efficient code [4]. We also made experiences with a Gaussian function, but the results did not justify the complexity added.

B.1 Scheduler Controller

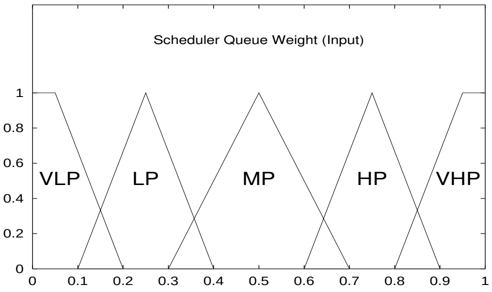

Fig. 2. Scheduler Membership functions: Queue Weight

The packet scheduler used in our architecture is WRR (Weighted Round Robin). In this scheduler, queues are served according to a configurable weight that can be changed during network operation. This allows to have control of the bandwidth assigned to each service class. The packet delay and discard rate for each queue (class) can be controlled by changing this weight. An example of membership function of schedule controller is showed in figure 2. Other membership functions are: packet delay in the EF queue and discard rate due to queue overflow in the BE class. The output membership functions are also defined as trapezoid functions by the same previous reasons. We used the center of gravity defuzzification method, since it gives better results. The output membership function gives the weights assigned to each class in the WRR scheduler.

B.2 Policer Controller

The objective of this controller is to reconfigure the policer, which is implemented by a token bucket. Thus, the first input variable in the controller is the number of tokens stored into the bucket. The second is the number of EF packets discarded in the policer. The third variable is the maximum delay of EF packets inside the domain that indicates the reduction in ingress traffic, reducing token bucket rate on policers in edge nodes.

The policer, however, cannot reduce the value of the bucket rate since this violates an agreement. On the other hand, packets in the EF class should not be discarded inside the domain. If there are no more resources in the domain core, i.e. delays are high, one should indicate a reduction of input rate in edge nodes, reducing the bucket rate.

B.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.

B.3.a Scheduler Controller. A synthesis of the scheduler controller rule base is presented below:
1. If the delay in EF queue is medium, then the queue weight is increased by one level; e.g., if the weight was low it goes to medium. And if EF delay is high the queue weight is increased by two levels.
2. If the delay in EF queue is low and the packet discard rate in BE queue is medium, then queue weight is reduced by one level. And if BE packet discard rate is high the queue weight is reduced by two levels.

B.3.b Policer Controller. A synthesis of the policer controller rule base is presented below:
1. If the EF queue delay in core node is medium, then the policer rate is reduced by one level. If EF queue delay in core nodes is high the policer rate is reduced by two levels.
2. If the packet discard rate of BE class in edge node is high and EF queue delay in core nodes is low, then the policer rate is increased by one level. If packet discard rate of BE class in edge node is high and EF queue delay is low, the policer rate is increased by two levels.

C. Optimization

The fuzzy logic allows the representation of ambiguous values and produces a correct answer even with uncertain inputs. However, an efficient behavior requests the definition of coherent rules. A methodology is desired to produce coherent and efficient membership functions and a correct rule base.

To improve the controller efficiency, we used optimization procedures. For rule creation we used the Wang-Mendel algorithm that constructs a group of coherent rules according to the desired behavior. For the membership functions optimization we used a genetic algorithm in order to find the best parameters combination.

C.1 Rule creation with the Wang-Mendel algorithm

The objective of this method is to create a group of rules according to a knowledge base. The Wang-Mendel method is shown in [20]. The basic steps of the algorithm are:

1. Starting with the first value, it creates a rule of the form IF ¡var¿ IS ¡adj¿ AND ¡var¿ IS ¡adj¿ AND . . . THEN ¡var¿ IS ¡adj¿.
2. If this rule does not exist and it does not contradict any other rule, it is added to the rule base.
3. If the rule already exists in the rule base it is ignored; however, its rank is increased.
4. If the rule contradicts some existent rule it is included in the rule base, however with a contradiction mark and a rank counter is started. At the end of the evaluation the contradictory rule with higher rank is included in rule base and the rule with lowest rank is discarded.

This methodology is useful to verify the consistence of the rule base, avoiding the occurrence of contradictory rules, which produce wrong results.

C.2 Genetic algorithm optimization

The genetic algorithm is based on Darwinian selection. The use of genetic algorithm to improve fuzzy parameters was proposed by Velasco [21], Cordón [22] and Herrera [6].

The genetic algorithm starts by creating a set of random solutions (called *individual*) to the problem. The set of individuals is called *population*. A score for each individual is found by testing the result of the fuzzy controller against a desired value. Some individuals with low score are replaced by new individuals created from individuals with high score. This process continues until a solution to the problem is found or some other end condition (e.g., elapsed time or number of generations) is reached. The individual in the population with the highest score is the solution to the problem.

There are several ways for creating a new individual from existing individuals. One method is the mutation. A single parent is selected, and the child is identical to the parent except some small random change. If an individual is represented by a bit string, mutation is often implemented by changing the value of a random bit from zero to one or from one to zero. Another reproduction method is crossover: two parents are selected, and the new individual is created by mixing their genes. For example, if an individual is represented by a fixed-length string of characters, a crossover point is chosen. The characters before the crossover point are selected from the first individual, and the characters after the crossover point are taken from the second parent.

Our scheduler fuzzy controller provides the WRR weight, that is not the actual function to be optimized. The variable that will be optimized is the edge-to-edge delay, produced only after a simulation. Then we use a simulation trace with controller parameters and actual delay as a knowledge base to the genetic algorithm. After a series of interactions we obtain a set of parameters that optimizes the controller.

IV. Simulation Model

The simulation model consists of EF (Expedited Forwarding) and BE (Best Effort) classes sharing bandwidth on a DiffServ domain. The EF cllas is suitable for real-time applications, like voice over IP, as it offers low delay and jitter, and assured bandwidth. The BE class does not offer any guarantee and it is likely to transport the most part of the traffic of IP networks. It is used to compete with the EF class traffic.

A. *Simulation Environment*

The platform used was the Network Simulator (NS), version 2.1b8 [23]. The fuzzy library used in this work was developed with the JFS tool from Mortensen [24]. This tool provides an interface for prototype development (specification of membership functions, inference rules and the defuzzificator), an initial verification of the model and optimization tools. The C code generated by this tool is compiled and linked to the NS.

B. *Simulation Scenario*

The voice over IP application is modeled with CBR and exponential On/Off traffic over UDP. Voice traffic is classified into EF class and the competing traffic, also CBR/UDP, is classified into BE class. The topology used is shown in figure 3. It is a DS domain composed of 40 nodes, where 30 core nodes (nodes 0 to 29) and 10 edge nodes (nodes 30 to 39). There are 5 ingress edge nodes (nodes 30 to 34) and 5 egress edge nodes (nodes 35 to 39). This topology was created with the gt-itm package (bundled in NS) [23].

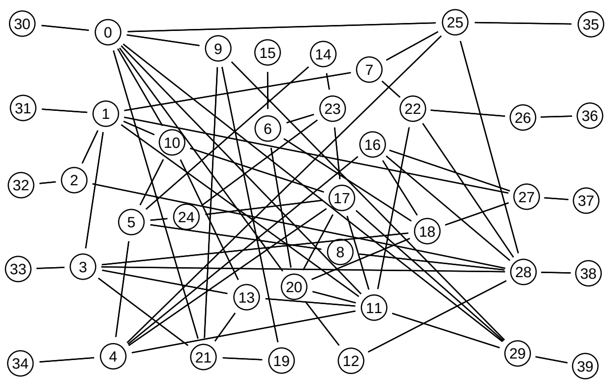

Fig. 3. Simulation Topology

The number of voice traffic sources (EF class) changes during the simulation time of 30 minutes. This behavior forces the controller to operate into a real situation. The number of voice sources varies according to an exponential distribution with the following active/inactive periods given in seconds: (120,60), (60,30), (180,60), (60,20).

Each CBR voice generates a 64 Kbps traffic rate (PCM channel). This does not include the IP overhead, which gives an actual rate of about 80 Kbps. In On/Off source, we use 64 Kbps peak rate with burst time of 400 ms and idle time of 600 ms, which gives an average rate of 25,6 Kbps. The size of the packet is 576 bytes in both cases. Using the CBR traffic, the number of sources varies from 0 to 150 active sources. In the On/Off traffic scenario, the number of sources varies from 0 to 300 active sources. In both scenarios, we use 150 and 300 competing 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 50 packets, which gives a maximum delay of 100 ms in each node. The simulation model uses a WRR scheduler, Drop Tail queues in both classes and Token Bucket policer for EF class (the BE class is not conditioned).

C. *Conventional controller*

With the aim to compare and validate our approach, a conventional digital controller was defined. With the input traffic scenario used here, a domain without any controller is clearly worst than the one with a 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 the queue weight in the scheduler, which increases 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 then applied to the output queue weight in the scheduler.

This controller uses the same sample period used on the fuzzy controller.

V. RESULTS

In this performance evaluation the following performance metrics were evaluated: percentile of edge-to-edge delay and jitter in EF class and discard rates in the EF and BE classes. For each evaluation, we used CBR and exponential On/Off traffic. We show the tables of the DiffServ domain without controller, with a conventional controller and with the proposed fuzzy controller. All simulations start with initial scheduler configuration with 50% of the bandwidth for each class. To eliminate simulation results with an empty network, we start collecting results 30 seconds (*warmup time*) after the beginning of the simulation.

A. *Result of the Optimization*

After optimization procedure was executed, we could verify the result comparing EF traffic delay, with CBR and On/Off traffics. Figure 4 shows the graph of edge-to-edge delay of a EF flow using CBR traffic. It compares the original fuzzy controller (without optimization) and the optimized fuzzy controller (after optimization with genetic algorithm). Figure 4 shows the graph of edge-to-edge delay using On/Off traffic, comparing original and optimized fuzzy controller. Both graphs show an improvement in QoS, as we expected.

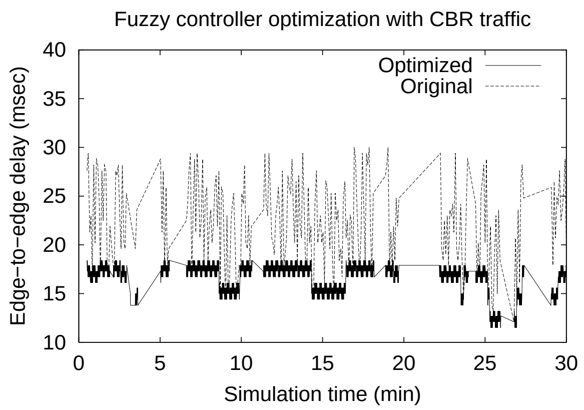

Fig. 4. Optimization of fuzzy controller with CBR traffic

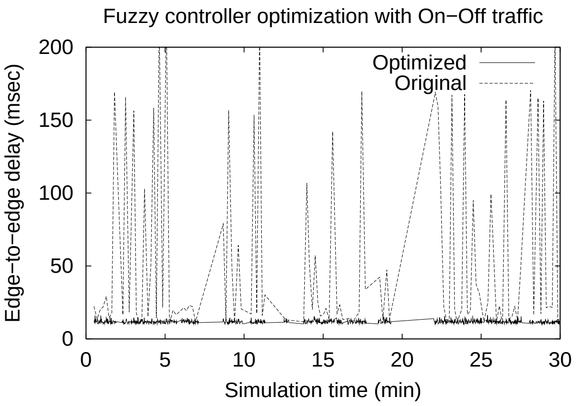

Fig. 5. Optimization of fuzzy controller with On/Off traffic

B. Edge-to-edge delay and jitter in the EF class

We show the simulation results by evaluating the percentile of delay and jitter for EF flows. We compare these metrics for the optimized fuzzy controler, the conventional controller and a DS architecture with a static configuration. Table I and II shows the delay and jitter of CBR and On/Off traffic in domain.

TABLE I
PERCENTILE OF EF DELAY ($msec$)

Traffic	Avg.	Per 50	Per 90	Per 95
CBR Without	198.7	201.5	239.0	249.9
CBR Conventional	43.9	43.1	65.0	70.2
CBR Fuzzy	33.2	32.2	52.4	56.4
OO Without	159.3	151.4	182.3	184.2
OO Conventional	23.4	21.6	38.5	47.7
OO Fuzzy	9.2	8.9	11.8	12.5

TABLE II
PERCENTILE OF EF JITTER ($msec$)

Traffic	Avg.	Per 50	Per 90	Per 95
CBR Without	19.8	0.0	71.9	72.1
CBR Conventional	1.7	0.0	0.0	0.0
CBR Fuzzy	1.0	0.0	0.0	0.0
OO Without	13.8	6.8	34.2	36.4
OO Conventional	12.0	9.0	26.9	35.5
OO Fuzzy	2.8	2.2	5.0	5.9

C. Discard rate on the DiffServ Domain

Table III shows the packet discard of the voice traffic in the EF class. We may notice a decrease of the EF discard with fuzzy controller for both traffics comparing the experiment with conventional controller and without any controller. Table IV shows the packet discard on default traffic in the BE class. In this situation the BE drops increase with fuzzy controller. We can see that fuzzy controller gives better results evaluating On/Off traffic.

TABLE III
EF DROPS IN DOMAIN

Controller	CBR	On/Off
Without	669797	3899233
Conventional	2138	54741
Fuzzy	15	415

TABLE IV
BE DROPS IN DOMAIN

Controller	CBR	On/Off
Without	2446937	5040731
Conventional	3114725	6442433
Fuzzy	3116753	6462221

VI. CONCLUSION AND FUTURE WORKS

In this paper, we showed a methodology for dynamic resource provisioning in a DiffServ architecture. The use of fuzzy logic improves the handling of inaccuracy and uncertainties of the ingress traffic into the domain.

The use of optimization tools, as Wang-Mendel and genetic algorithms, give us better controller parameters, independently of the designer ability. The use of actual configuration parameters, got during simulation as knowledge base for the genetic algorithm gives a good optimization. This produces better results even for different topologies and traffic patterns. The controller has a low complexity, which maintais DiffServ scalability characteristic.

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.

REFERENCES

[1] M. P. Fernandez, A. de Castro P. Pedroza, and J. F. de Rezende, “Qos provisioning across a diffserv domain using policy-based management,” in *(Globecom 2001)*, (San Antonio, USA), Nov. 2001.

[2] S. Blake, D. Black, and M. Carlson, “An architecture for differentiated services.” RFC 2475, Dec. 1998.

[3] M. P. Fernandez, A. de Castro P. Pedroza, and J. F. de Rezende, “Quality of service in a diffserv domain using policy-based management,” in *XVII International Teletraffic Congress (ITC’17)*, (Salvador, Brasil), Sept. 2001.

[4] C. C. Lee, “Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part II,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, pp. 419–435, Mar. 1990.

[5] D. Lorenz and G. Orda, “Qos routing in networks with uncertain parameters,” in *IEEE Infocom 98*, 1998.

[6] F. Herrera, M. Lozano, and J. Verdegay, “Tuning fuzzy logic controllers by genetic algorithms,” *International Journal of Approximate Reasoning*, vol. 12, pp. 299–315, June 1995.

[7] J. Kim, Y. Moon, and B. P. Zeigler, “Designing fuzzy net controllers using GA optimization,” in *Proceedings IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, (Tucson (AZ), USA), pp. 83–88, Mar. 1994.

[8] P. Trimintzios, G. Pavlou, I. Andrikopoulos, D. Griffin, C. Jacquenet, P. Georgatsos, Y. T’joens, L. Georgiadis, R. Egan, and G. Memenios, “An architectural framework for providing qos in ip differential service networks,” in *VII IFIP/IEEE International Symposium on Integrated Network Management (IM 2001)*, 2001.

[9] G. Pavlou, I. Andrikopoulos, D. Griffin, C. Jacquenet, P. Georgatsos, Y. T’joens, L. Georgiadis, R. Egan, and G. Memenios, “Service level specification semantics, parameters and negotiation requirements.” Internet Draft draft-tequila-diffserv-sls-00.txt, July 2001.

[10] P. Flegkas, P. Trimintzios, G. Pavlou, I. Andrikopoulos, D. Griffin, C. Jacquenet, P. Georgatsos, Y. T’joens, L. Georgiadis, R. Egan, and

G. Memenios, "On policy-based extensible hierarchical network management in qos-enable ip networks," in *Workshop on Policies for Distributed Systems and Network (Policy 2001)*, (Bristol - UK), Jan. 2001.

[11] L. Westberg, M. Jacobsson, G. Karagiannis, and S. Oosthoek, "Resource management in diffserv (rmd) framework." Internet Draft draft-westberg-rmd-framework-00.txt, Apr. 2001.

[12] L. Westberg, M. Jacobsson, G. Karagiannis, and S. Oosthoek, "Resource management in diffserv on demand (roda) phr." Internet Draft draft-westberg-rmd-od-phr-00.txt, Apr. 2001.

[13] R. R. Liao and A. T. Campbell, "Dynamic edge provisioning for core networks," in *IFIP/IEEE Eighth International Workshop on Quality of Service (IWQoS 2000)*, (Pittsburgh - USA), June 2000.

[14] R. R. Liao and A. T. Campbell, "Dynamic core provisioning for quantitative differentiated service," in *IFIP/IEEE Ninth International Workshop on Quality of Service (IWQoS 2001)*, (Karlsruhe - Germany), June 2001.

[15] S. Ghosh, Q. Razouqi, H. J. Schmacher, and A. Celmins, "A survey of recent advances in fuzzy logic in telecomunications networks and new challenges," *IEEE Transactions on Fuzzy Systems*, vol. 6, pp. 443–447, Aug. 1998.

[16] B. Li and K. Nahrstedt, "A control-based middleware framework for quality of service adaptions," *IEEE Journal on Selected Areas in Communications*, Sept. 1997.

[17] R. Cheng and C. Chang, "Design of a fuzzy traffic controler for atm networks," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 460–469, June 1996.

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

[19] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[20] L. X. Wang and J. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, pp. 1414–1427, July 1992.

[21] J. Velasco and L. Magdalena, "Genetic algorithms in fuzzy control systems," in *Genetic Algorithms in Engineering and Computer Science* (G. Winter, J. Periaux, M. Galan, and P. Cuesta, eds.), pp. 141–165, John Wiley & Sons, 1995.

[22] O. Cordón, F. Herrera, and A. Peregrín, "Looking for the best defuzzification method features for each implication operator to design accurate fuzzy models," tech. rep., University of Granada, Apr. 1999. Technical Report DECSAI-99108 , Dept. of Computer Science and A.I., University of Granada.

[23] "Network simulator - ns version 2." http://www.isi.edu/nsnam/ns/, 1998.

[24] J. E. Mortensen, "Jfs fuzzy software." http://www.inet.uni2.dk/ jemor/jfs.htm, 1998.