Quality of Service control in IP Networks using Fuzzy Logic for Policy Condition Evaluation

Raulison A. Resende[1], Nadia A. Nassif[1], Marcos A. de Siqueira[1],
Ademilson E. da Silva[1], Mamede Lima-Marques[2]

[1]**CPqD** Telecomunications R&D Center
Rodovia Campinas Mogi-Mirim, km 118,5 - 13086-902 Campinas - SP- Brazil

{raulison, nadia, siqueira, ademilson}@cpqd.com.br

[2]University of Brasilia - **UnB**

mamede@unb.br

Abstract- This paper presents the architecture of a policy-based network management system designed specifically for Quality of Service management, where high level business policies are modeled as ECA (*Event-Condition-Action*) rules. Specifically, the system is mainly based on the policy architecture being proposed by the IETF (*Internet Engineering Task Force*), DMTF (*Distributed Management Task Force*), and TMF (*TeleManagement Forum*) standardization bodies. This work proposes a novel approach for policy condition evaluation using fuzzy logic. The fuzzy controller has the ability to examine policy conditions differently from default condition analyzers that employ simple conditions formed by a set of (IFs, ANDs and ORs), allowing the consideration and correlation of several input variables before taking decisions for the execution of policy actions. The system have been implemented and evaluated over a test bed network formed by Cisco® routers.

***Key-words*: Policy-Based Network Management, Quality of Service, Fuzzy Logic.**

I. INTRODUCTION

Over the last decade, academia and industry have expended considerable effort researching QoS mechanisms for IP networks. Several solutions have been developed, and some of which are currently implemented in ISPs (*Internet Service Providers*) and corporate networks. Within this context, remarkable technologies can be listed, including traffic queuing strategies such as CBQ (*Class-based Queuing*), WFQ (*Weighted Fair Queuing*), PQ (*Priority Queuing*) and others; QoS architectures such as IntServ (*Integrated Services Architecture*) [1] and DiffServ [2]; congestion avoidance strategies such as RED (*Random Early Discard*) and traffic engineering using MPLS (*Multiprotocol Label Switching*) [3].

The heterogeneity of the technological options for QoS deployment has lead to an extremely complex scenario for the interoperation and customization of QoS. As a result, in practice, only few mechanisms cited above have been deployed in IP networks.

Meanwhile, the demand for implementing QoS mechanisms continues to grow. The increased transport of real time and mission critical applications over IP-based networks, added to the explosion of P2P (*Peer to Peer*) traffic, and provision of new services such as BGP/MPLS based VPNs (*Virtual Private Networks*) [4], require the implementation of control mechanisms for differentiated traffic forwarding priorities in routers and switches. MPLS based VPNs service, nowadays offered by most of the ISPs (*Internet Service Providers*), as a low cost alternative to traditional layer 2 VPNs such as Frame Relay and ATM. Applying QoS mechanisms as DiffServ with MPLS based VPNs, it is possible to differentiate traffic within an LSP (*Label Switching Path*) , thus allowing clients to determine, for instance, that VoIP (*Voice over IP*) traffic should have increased priority over the rest of the traffic on the ISP backbone.

QoS mechanisms currently implemented in real networks, such as classifiers, schedulers and markers, are normally configured statically in network devices. Therefore, there is no ability for these mechanisms to respond to changes in traffic and the network state. Furthermore, mapping enterprise policies into the appropriate configuration parameters for QoS mechanisms is generally performed manually by the network administrator.

On the other hand, at the network management area, a new architecture called PBNM (*Policy-Based Network Management*) have emerged allowing high-level business policies to be translated into appropriate configuration of network devices by the management systems, thus reducing the complexity of the network operation, independent of the type, model, vendor and operating system used. This architecture has emerged allowing some degree of automated operation of networks and the adaptation of the configuration of network devices to provide suitable QoS (*Quality of Service*) for different traffic flows or classes of service

This paper briefly describes the design and implementation of a PBNM system that performs the complete cycle of policy creation, validation, enforcement and monitoring. Network events are monitored and a policy-based automatic re-configuration of network devices is performed. The re-configuration is performed based on the result of a condition analysis. The duties of the network administrator are related solely to the creation of business policies using a Policy Editor. The PBNM system is responsible to map high-level business policies into device specific configurations in order to enforce the company goals.

Specifically, this paper is focused on the module responsible for the condition analysis. A Fuzzy-based policy condition analyzer is implemented through the association of network state information, policy events and conditions with Fuzzyfied curves. This strategy allows policy decision to be performed based on Fuzzy policies, including the analysis of several conditions at the same time.

The paper is organized as follows. Section II describes the policy events, conditions and actions for QoS management, Section III presents the overall system architecture; Section IV details the Fuzzy condition analyzer module implementation; and finally, Section V presents the validation of the Fuzzy-based

policy condition analyzer over a real test bed network composed by Cisco® routers.

II. QoS Policies

One of the biggest challenges in a PBNM system is the creation of new types of policies at runtime, without changing hard coded structures and functions. In most of the PBNM systems available today, a limited number of different types of policies can be instantiated. Our system was designed such that the network administrator would be able to create customized policies through the selection of a set of events, conditions and actions. As such, a small number of types of events, conditions and actions can be combined to create a large number of different policies.

For this first version of the system, we focused on managing DiffServ mechanisms for corporate routers. In this scenario, the possible events, conditions and actions for creating the QoS policies are presented below:

Policy Events

- *Specific Trap:* this event is generated when the management system receives a specific SNMP TRAP (*Simple Network Management Protocol*) from a given network element;
- *Scheduled event*: this event is generated by a scheduler. The specific time, or time interval for event generation are configurable attributes of the event;
- *Threshold*: this event is generated by a monitor to notify that a predefined QoS performance threshold has been reached. The monitor should be installed in accordance with a given policy activation.

Event filters and event monitors should be installed in accordance with specific policy events and should be auto configurable so as to receive only the events that are relevant to the current active policies. The system should be designed in such a way that new kinds of events can only be added to the system by inserting the relevant XML (*eXtensible Markup Language*) files describing the event information model into the policy repository.

Simple Policy Conditions

Simple policy conditions are represented in the DEN-ng policy model [6] as an aggregation of a set of *PolicyStatements*. Their evaluation consists of a set of ANDs of ORs, or vice versa, comparing *PolicyVariables* with *PolicyValues* using the *PolicyOperators*

The simple policy conditions to be implemented in our system are listed below:

- Is the network congested?
- Is it necessary to add a congestion prevention mechanism?
- Is the current set of classes overloaded?
- Is the current set of classes under-loaded?

In our system there is a monitor responsible for collecting relevant information about network performance. The information collected is stored in a data base and is used when the system need to take a decision as a result of a condition analysis.

Fuzzy Policy Conditions

The fuzzy conditions are capable of correlating several parameters such as link occupation, packet loss, delay, jitter and the time of day to give a numerical output of bandwidth to be reserved for a given traffic class. The fuzzy logic-based condition analysis is better described in section IV.

Similarly to the simple conditions, there is a monitor responsible for collecting QoS performance parameters from the network. These parameters are used as an input for the fuzzy logic-based condition analysis.

Policy Actions

Specific policy actions shall be executed after policy condition evaluation. For instance, action "A" would be executed if the analysis condition result were TRUE, action "B" if the analysis condition result were FALSE or action "C" as a result from Fuzzy condition analysis. The association of which actions may be executed and the occurrence of an event and a condition analysis result is performed by the network administrator at the policy editor.

Below are listed the policy actions for configuring DiffServ parameters.

- Classifier configuration;
- Marker configuration;
- Traffic Shaping or Policing configuration;
- Dropper configuration;
- Queue scheduler configuration;

We adopted the strategy of using an object-oriented information model for representing the events, conditions, actions and their relationships. In order to contextualize the FCA (*Fuzzy Condition Analyzer*) module, the next section briefly presents the system implementation architecture, described with details by Siqueira et al. [7].

III. QoSM Implementation architecture

This section presents the implementation architecture for the QoS Policy Manager (*QoSM*). The system is composed of internal elements responsible for policy edition, compilation, conflict detection and resolution, and policy evaluation. There are interfaces to external elements such as the policy repository, network configuration and monitoring proxies.

The work of standardization bodies, such as the IETF and DMTF, in relation to PBNM, has mostly focused on the architecture of policy-based systems, information models and protocols. Conversely, DEN-ng policy model is one step forward in this respect, specifying the policy life cycle, some internal components such as a policy state machine, and it also presents some use cases for policy evaluation.

Based on the DEN-ng approach, we propose the PDP implementation architecture that is illustrated in Fig. 1.

QoSM Interfaces:

- I-PMT (*Interface to the Policy Management Tool*): provides web-based access for network administrators to insert, edit and delete policies;
- I-PR (*Interface to the Policy Repository*): provides access to the policies (events, conditions, actions and their relationships) stored in the policy repository. The Policy Repository stores the network policies that will be analyzed by the QoSM in order to make decisions and translate them into network configuration modifications. In our system, the Policy Repository was modeled as an extension of the DEN-ng policy model for managing QoS mechanisms;

- I-PSMR (*Interface to the Policy State Machine Repository*): provides access to the policies state repository. The Policy State Machine repository maintains the execution state of all active policies. Valid states are: event not occurred; event occurred; conditions satisfied; conditions not satisfied; action executed (sent to the device); successful action execution at the device; and some other combinations of different states;
- I-NELC (*Interface to the Network Element Layer Configuration Proxy*): provides access to the proxies responsible for the configuration of the network devices. This interface may receive policy actions modeled in XML from the AG module (described below) and translate them into specific commands for the devices being managed. This project contemplates the implementation of a proxy that supports CLI (*Command Line Interface*) configuration;
- I-NELM (*Interface to the Network Element Layer Monitoring Proxy*): provides access to the monitoring proxies. The proxies may access devices via SNMP to monitor network and traffic states.

QoSM internal modules:

- PE (*Policy Editor*): implements the server side presentation logic of the user web-based policy editor. The policy editor presents a set of events, conditions and actions to the user. The policy is created through the association of a set of events, conditions and actions. Furthermore, the PE enables the configuration of specific attributes of each policy component, such as threshold values for events, limits for condition variables and configuration parameters for policy actions;
- PC (*Policy Compiler*): translates the policies from the business view to the Policy Information Model format. Specifically in this project, the policies to be stored at the policy repository are translated into object-orientated objects according to the DEN-ng model;
- CD (*Conflict Detector*): enables the detection of conflicts, both intra policy and inter policy;
- EG (*Event Generator*): this module generates specific pre-configured events, mostly obtained via the I-NELM interface and subsequently used to trigger the analysis of the set of conditions of the policies associated with the given event;
- CA (*Condition Analyzer*): provides a simple condition analysis (using the operators IF, AND and OR);
- FCA (*Fuzzy Condition Analyzer*): provides an advanced condition analysis, as further described in Section IV;
- AG (*Action Generator*): sends specific actions to the I-NELC interface by means of the condition evaluation of TRUE or FALSE;
- SMC (*State Machine Controller*): communicates with the EG, CA and AG modules. It updates the policy state data, enabling the system to avoid policy evaluation errors and assisting the network administrator to monitor the levels of QoS levels applied through the policies.

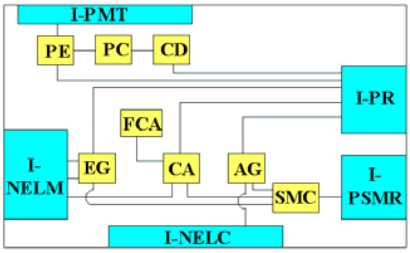

Fig. 1 - QoS Manager internal architecture.

IV. FUZZY CONDITION ANALYZER (FCA) ARCHITECTURE

Much work has been done on Policy Management. Nevertheless, there are some issues still open for research, such as policy conflict detection, policy evaluation strategies, high level policy representation and translation, policy representation language, and others.

Little attention has been dedicated to policy condition evaluation strategies. Most of the implementations use a set of (ANDs of ORs) or a set of (ORs of ANDs) for policy condition evaluation.

This scheme leads to a very predicable decision strategy. We propose in this work a different approach for policy condition evaluation using Fuzzy logic. The FCA module is formed by a Fuzzy controller able to take several variables from the network, compare them using fuzzyfied curves and give a result for the condition.

Following we describe how the FCA was implemented in Java, J2SE v1.4 with Postgres Database.

A. Fuzzy Controller

Fuzzy Logic is also being used to control large engineering plants such as power stations, effluent treatment systems and even to maintain a steady flight path for pilot less reconnaissance aircraft. By defining a method that allows a mathematical description of vague, imprecise and possibly conflicting information, sophisticated systems may be devised to significantly improve system performance. Fuzzy controllers are conceptually very simple. They consist of an input stage (fuzzyfication), a processing stage (inference), and an output stage (defuzzification). The input stage maps sensor or other inputs, such as switches, thumbwheels, and so on, to the appropriate membership functions and truth-values. The processing stage invokes each appropriate rule and generates a result for each, then combines the results of the rules. Finally, the output stage converts the combined result back into a specific control output value. Fuzzy logic provides a simple and powerful method of decision-making that facilitates a link between basic logic and practical applications. It has been shown that a fuzzy logic controller can provide algorithms which convert the linguistic control strategies based on intuition, heuristic learning and expert knowledge into an automatic control strategy for bandwidth management purposes [10] [11] and [14]. A fuzzy inference system is usually rule based and has the inherent capability to perform inference and derive results with imprecise models and quantities. The block diagram of a general fuzzy controller is illustrated in Fig. 2.

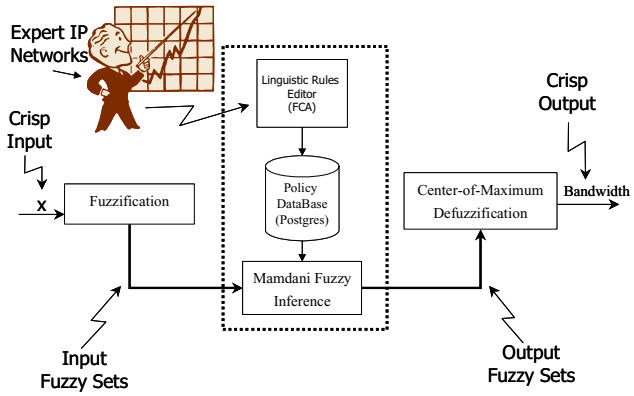

Fig. 2 - Block diagram of a fuzzy inference system

1) Membership Functions to the Fuzzy Controller

The fuzzy membership function used in the solution of our problem is continuo function: triangular trimf(x; a, b, c), as described in the equation below(1).

$$trimf(x;a,b,c) = \begin{cases} 0, & if \quad x \le a \\ \dfrac{x-a}{b-a}, & if \quad x \in (a,b] \\ \dfrac{c-x}{c-b}, & if \quad x \in [b,c) \\ 0, & if \quad x \ge c, \end{cases} \qquad (1)$$

where b is modal value, and a and c denote the lower and upper bounds, respectively, for nonzero values of trimf(x). The membership values were obtained by mapping the values obtained from the experiment developed in TIPHON (*Telecommunications and Internet Protocol Harmonization Over Networks*) project for Voice over IP (VoIP) quality [12].

B. Controller Variable

We have adopted the link occupation, end-to-end packet loss, end-to-end peak jitter, period of day and end-to-end delay as entry variables in our system, which give us a rate available in the system to assure quality of service.

The range of all possible values for an end-to-end link metric is as follows: "Link Occupation" presented in Table I, "End-to-End Packet loss" shown in Table II, "Peak Jitter" shown in Table III, "Period of Day" shown in Table IV, "End-to-End Delay" shown in Table V. The "Fuzzy Controller Bandwidth" can be seen in Table VI. The graphical representation of the membership functions for each one of the variables is presented in the Figures 3, 4, 5, 6 and 7 respectively.

TABLE I
FUZZY MEMBERSHIP FOR FUZZY INPUTS
"LINK OCCUPATION" - ***LO***

Linguistic Variable	Type Function	Parameters Function (a,b,c)	Values
LOW	trimf	(x; L_a, L_b, L_c)	(x;0,32,96)
MEDIUM	trimf	(x; M_a, M_b, M_c)	(x; 64,128,192)
HIGH	trimf	(x; H_a, H_b, H_c)	(x;160,224,256)

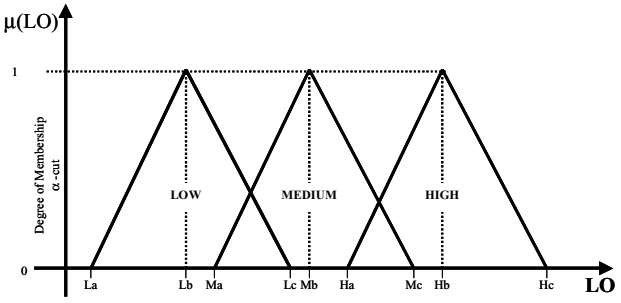

Fig. 3 - Membership functions of the linguistic variable (input) "Link Occupation" - ***LO***

TABLE II
FUZZY MEMBERSHIP FOR FUZZY INPUTS
"END-TO-END PACKET LOSS RATIO %" - ***PL***

Linguistic Variable	Type Function	Parameters Function (a,b,c)	Values
GOOD	trimf	(x; G_a, G_b, G_c)	(x; 2, 6, 10)
MEDIUM	trimf	(x; M_a, M_b, M_c)	(x; 7, 14, 21)
POOR	trimf	(x; P_a, P_b, P_c)	(x; 17, 25, 33)

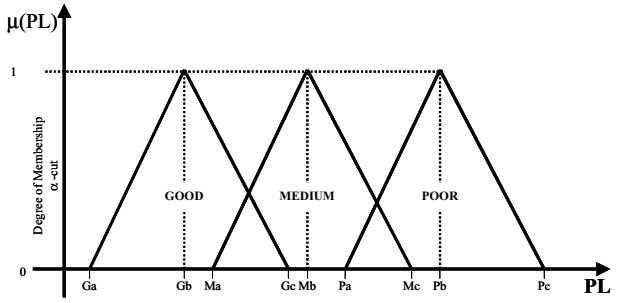

Fig. 4 – Membership functions of the linguistic variable (input) "End-to-End Packet loss ratio %" - ***PL***

TABLE III
FUZZY MEMBERSHIP FOR FUZZY INPUTS
"END-TO-END PEAK JITTER (DELAY VARIATION)" - ***PJ***

Linguistic Variable	Type Function	Parameters Function (a,b,c)	Values
GOOD	trimf	(x; G_a, G_b, G_c)	(x; 50, 75, 100)
MEDIUM	trimf	(x; M_a, M_b, M_c)	(x; 84, 125, 166)
POOR	trimf	(x; P_a, P_b, P_c)	(x; 151, 225, 299)

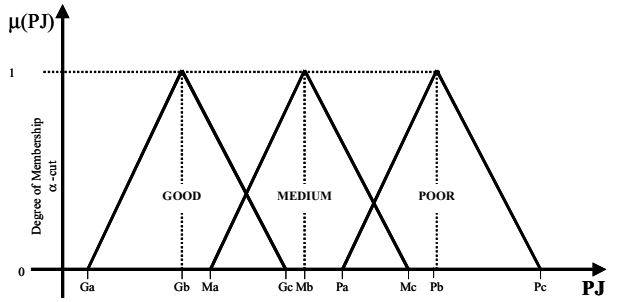

Fig. 5 – Membership functions of the linguistic variable (input) "End-to-End Peak Jitter" - ***PJ***

TABLE IV
FUZZY MEMBERSHIP FOR FUZZY INPUTS
"PERIOD OF DAY" - ***PD***

Linguistic Variable	Type Function	Parameters Function (a,b,c)	Values
DAWN	trimf	(x; D_a, D_b, D_c)	(x; 0, 3, 6)
MORNING	trimf	(x; M_a, M_b, M_c)	(x; 5, 9, 12)
AFTERNOON	trimf	(x; A_a, A_b, A_c)	(x; 11, 15, 18)
NIGHT	trimf	(x; N_a, N_b, N_c)	(x; 17, 21, 24)

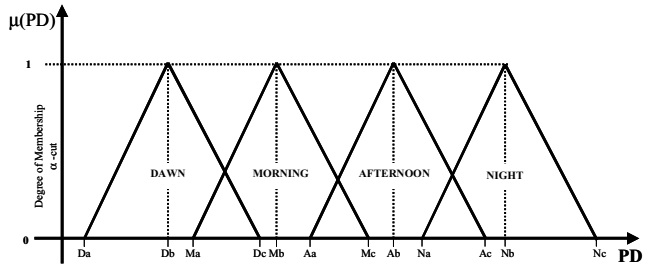

Fig. 6 – Membership functions of the linguistic variable (input) "Period of Day" - ***PD***

TABLE V
FUZZY MEMBERSHIP FOR FUZZY INPUTS
"END-TO-END DELAY (TRANSFER DELAY)" - ***TD***

Linguistic Variable	Type Function	Parameters Function (a,b,c)	Values
BEST	trimf	(x; B_a, B_b, B_c)	(x; 101, 150, 200)
HIGH	trimf	(x; H_a, H_b, H_c)	(x; 168, 250, 333)
MEDIUM	trimf	(x; M_a, M_b, M_c)	(x; 235, 350, 466)
LOW	trimf	(x; L_a, L_b, L_c)	(x; 302, 450 599)

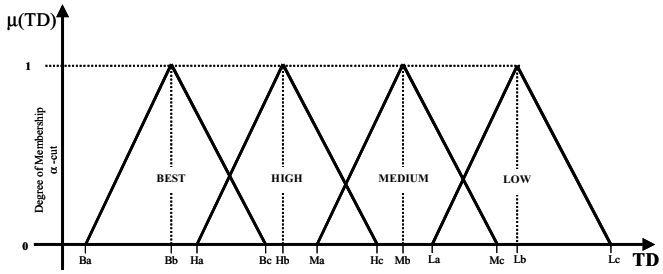

Fig. 7 – Membership functions of the linguistic variable (input) "End-to-End Delay (Transfer Delay)" - ***TD***

TABLE VI
FUZZY MEMBERSHIP FOR FUZZY OUTPUT
"FUZZY CONTROLLER BANDWIDTH" - ***CB***

Linguistic Variable	Type Function	Parameters Function (a,b,c)	Values
Low positive	trimf	(x; L_a, L_b, L_c)	(x;2,5,8)
Medium positive	trimf	(x; M_a, M_b, M_c)	(x;6,9,12)
High positive	trimf	(x; H_a, H_b, H_c)	(x,10,13,16;)
Low negative	trimf	(x; L_{na}, L_{nb}, L_{nc})	(x;-2,-5,-8)
Medium negative	trimf	(x; M_{na}, M_{nb}, M_{nc})	(x;-6,-9,-12)
High negative	trimf	(x; H_{na}, H_{nb}, H_{nc})	(x; -10,-13,-16)

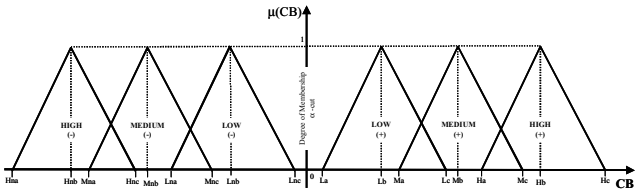

Fig. 8 – Membership functions of the linguistic variable (output) "Fuzzy Controller Bandwidth" - ***CB***

C. *Basic Fuzzy Rules*

The acknowledgment that relates the several variables and their respective values was summarized in table IV, which originates the basic fuzzy rules with the If-then format. The inference process used in the simulation is known as Mamdani, proposed in 1976 by Ebrain Mamdani [9]. The output variable Bandwidth has six terms and uses Center-of-Maximum (*CoM*) defuzzification. Thus, we can define some inference rules based in the table VII. For instance:

1. If (**Link Occupation** is **LOW**) And (**End-to-End Delay** is **MEDIUM**) And **(Period of Day** is **AFTERNOON)** And (End-to-End Peak Jitter is **MEDIUM**) And **(End-to-End Packet loss** is **POOR)** Then (**Fuzzy Controller Bandwidth** is **LOW Positive**)

As already described, fuzzy policy conditions were deployed taking as input Link occupation, end-to-end delay, period of day, end-to-end packet loss and current controller bandwidth. The FCA module gives as output the differential bandwidth that should be configured for a given link. Fig. 9 shows a screenshot of the fuzzy condition editor where the network administrator can create fuzzy conditions

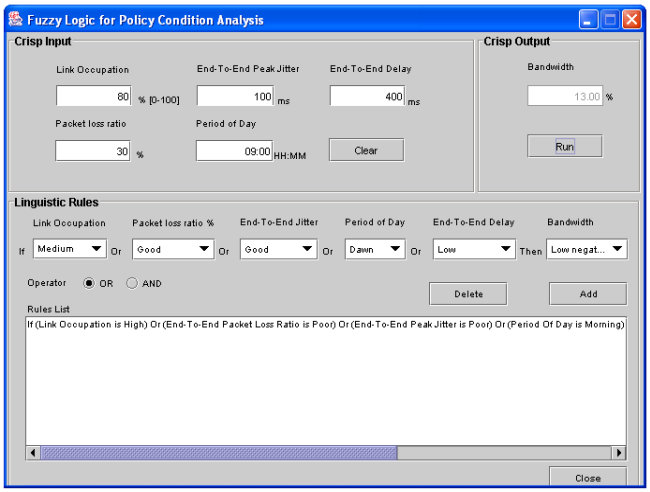

Fig. 9 – FCA policy condition editor

V. VALIDATION

Fig. 10 shows the testbed network where the QoS policy management system was applied. The network is formed by a Juniper M10 router (P1), three Cisco 3620 router (PE1, P2, and PE2) and a Cisco 7200 router (P3). A network tester (Agilent RouterTester) was used for traffic generation and measurement.

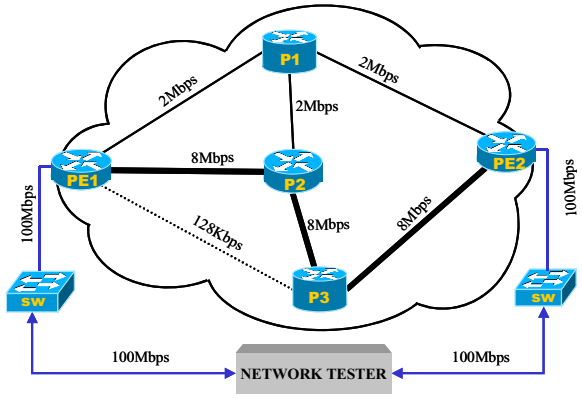

Fig. 10 – Testbed network

For a given set of rules configured at the FCA interface (shown in Fig. 9), based on the result of the execution of the fuzzy condition analyzer, some actions related to the policy were executed. The QoSM system (QoS Manager) generated the following configuration at the routers with the bandwidth percentage needed to be reserved for each class of service:

```
class-map match-all classe7
  match ip dscp 7
class-map match-all classe6
  match ip dscp 6
class-map match-all classe5
  match ip dscp 5
policy-map TesteQoS
  class classe7
   bandwidth percent 30
  class classe6
   bandwidth percent 25
  class classe5
   bandwidth percent 15
  class class-default
   bandwidth percent 5
Interface x/x
Service-policy TesteQoS
```

As can be inferred, the interfaces were configured with the CB-WFQ (Class-based Weighted Fair Queueing) scheduling algorithm, class 0 received 30% of bandwidth, class 1 received 25%, class 2 received 15% and class 3 received 5%. Fig. 11 shows the results for the RouterTester Over-Subscription-QoS test executed over the network configured automatically by the QoSM system.

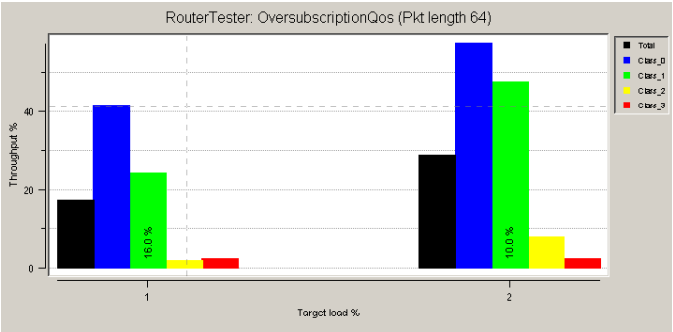

Fig. 11 – Test results

The left curves are for 16Mbps total throughput and the right curves are for 10Mbps. The traffic for each class is generated evenly. The network support at most 8Mbps.

As can be seen, the total throughput for each class is in accordance with the reserved values for each class.

VI. CONCLUSION

In this paper we have described the implementation architecture of a PBNM system designed specifically for managing QoS mechanisms in IP networks. The major contributions of our work are the introduction of new concepts into the architecture, such as a flexible policy editor that allows the composition of new types of policies, a fuzzy condition analyzer that has the ability to compare policy conditions differently from the default condition analyzers that use a set of IFs, a distributed event and configuration proxy scheme for collecting network events and configuring different types of network elements, the specialization of the CIM event model for QoS events, and the proposal of an implementation architecture for the PDP.

VII. REFERENCES

[1] R. Brader, D. Clark and S. Shenker, *Integrated Services in the Internet: an overview*, IETF RFC 1633, June 1994.

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

[3] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, *Requirements for Traffic Engineering Over MPLS*, RFC 2702, September 1999.

[4] E. C. Rosen and Y. Rekhter, *BGP/MPLS VPNs (RFC2547bis)*, IETF internet draft (work in progress), September 2003.

[5] Moore, B., Ellesson, E., Strassner, J. and A. Westerinen, "*Policy Core Information Model -- Version 1 Specification*", IETF RFC 3060, Fevereiro 2001

[6] J. Strassner, *Policy-Based Network Management, Solutions for the Next Generation*, Morgan Kaufmann, 2003.

[7] M. A. Siqueira, N. A. Nassif, R. A. Resende, A. E. da Silva, M. L. Marques, *Policy-Based Architecture for QoS Management in Enterprise IP Networks*, IEEE IM 2005.

[8] W. Pedrycz and F. Gomide, "*An Introduction to Fuzzy Sets: Analysis and Design*" MIT Press, pp. 8–10, 1998.

[9] E. H. Mamdani and S. Assilian, "*An experiment in linguistic synthesis with a fuzzy logic controller*," International Journal of Man Machine Studies, Vol. 7, No. 1, pp. 1-13, 1975.

[10] P. Siripongwutikorn, S. Banerjee, Tipper, D., "*Adaptive bandwidth control for efficient aggregate QoS provisioning*", GLOBECOM 2002 - Global Telecommunications Conference, Vol. 3, pp. 2435 – 2439, Nov./2002.

[11] Bin Qiu, "*Intelligent algorithms for QoS management in modern communication networks*", ICT 2003 - 10th International Conference on Telecommunications, Vol. 2, pp. 1489 – 1493, Feb./March 2003.

[12] TR 101 329: *Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General aspects of Quality of Service (QoS),* V1.2.5 (1998-10).

[13] M. P. Fernandez, et al., *QoS provisioning across a DiffServ domain using policy-based management*, IEEE GLOBECOM '01,Volume: 4, pp. 2220 – 2224, 25-29 Nov. 2001.

[14] R. A. Resende, A.Yamakami, S. M. Rossi, L. H. Bonani e E. Maschim, "*Traffic Engineering with MPLS Using Fuzzy Logic for Application in IP Networks*", FUZZ-IEEE 2003, St. Louis, MO - USA, May 25-28, 2003.