Service Level Agreement Enforcement for Differentiated Services

Paulo Rogério Pereira[1]

[1] INESC ID, Rua Alves Redol, 9. 1000-029 Lisboa, Portugal.
Phone: +351-213100345. Fax: +351-213145843. Email: prbp@inesc.pt

Abstract. This paper describes a hierarchical architecture of active policies that performs the management of a differentiated services (DiffServ) network. These policies monitor quality of service (QoS) parameters and dynamically optimize some aspects of the existing equipment and services to provide the best possible QoS to users. This helps the enforcement of a Service Level Agreement (SLA) between a provider and users. The results show that the use of active policies improves the service offered to users, by constantly adapting to the network state, and helping to fulfill SLAs with minimum costs to the service provider.

1 Introduction

Quality of Service (QoS) is defined as the collective effect of service performance which determines the degree of satisfaction of a user of the service [1]. Quality is measured through parameters such as service availability, delay, jitter, throughput and packet loss ratio [2].

The importance of QoS is such that many service providers offer services with a specified level of quality as defined in a Service Level Agreement (SLA) with users [2][3].

An important problem is how to evaluate and assure the QoS specified in the SLA. The QoS objectives specified in a SLA are high-level policy goals, sometimes not using the same parameters that can be measured or controlled on the network. Thus a refinement process is necessary to relate the different abstraction levels.

To address these problems, this paper introduces the concept of active policies [4] to help SLA fulfillment. Active policies are active objects which evaluate the QoS users are obtaining and dynamically optimize some aspects of the network operation at different abstraction levels, improving the QoS, whenever possible, up to the set goals.

The active policies are used in a scenario of a differentiated services (DiffServ) network [5] and the QoS improvements analyzed.

The paper is organized as follows. Section 2 describes the active policy model. Section 3 describes the scenario implemented, the active policies deployed and the simulation results. Section 4 compares our approach with related work. Finally, section 5 draws the conclusions and points further research topics.

2 Active Policies

Active policies are organized in a hierarchy where each active policy operates at a certain abstraction level, specialized on managing a certain problem over a subpart of the network. This division allows decomposing the SLA management problem in several simpler problems. The overall architecture of the active policies cooperates to enforce the SLA. Each active policy does a certain effort and relies on the effort of other lower-level active policies to achieve a specified QoS objective.

Figure 1 shows the internal modules of an active policy. Each active policy monitors some parameters of the managed objects under its responsibility and calculates the corresponding QoS parameters. A reactive management module provides quick responses to urgent problems. A proactive management module acts based on the history of past QoS to fine tune the operation of the reactive module and ensure a more precise SLA enforcement.

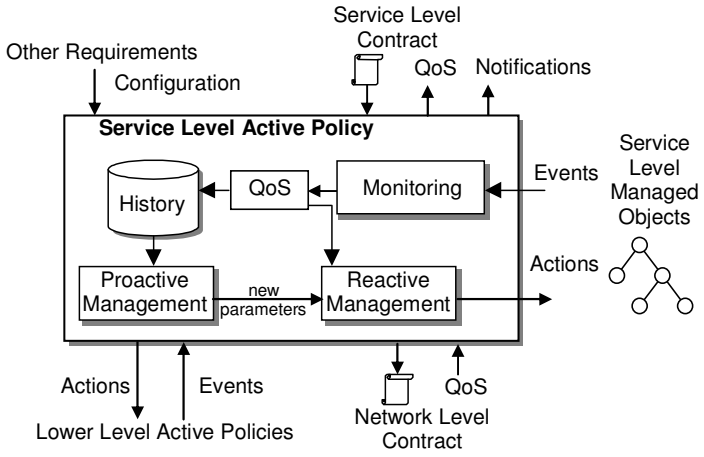

Fig. 1. Active policies internal modules

The active policies' monitoring module is shown in figure 2. For the monitoring operation, the first step is to sample the system parameters required for determining the QoS. The data is read from the managed objects or from other lower level policies. Then, it is converted to the required format and pre-processed, for instance with some low-pass function to filter peaks. Finally, the QoS parameters are calculated.

The reactive module, shown in figure 3, tries to fulfill an objective that is configured. The reactive module has a set of acting rules that fire the execution of some actions or the generation of events according to the QoS measured. The limiters verify if the actions make sense, limiting the modifications to what is allowed. The exception generation block verifies if there is a QoS degradation or if it not possible to act to fulfill the objectives. The appropriate exception actions are taken.

The proactive module, shown in figure 4, does planning tasks. Internally, it is similar to the reactive module, but acts according to the evolution of the observed QoS, instead of depending directly on the QoS. Additionally, the proactive module, instead

of acting over the managed objects, can reconfigure the policies themselves. This feedback provides more precise QoS fulfillment.

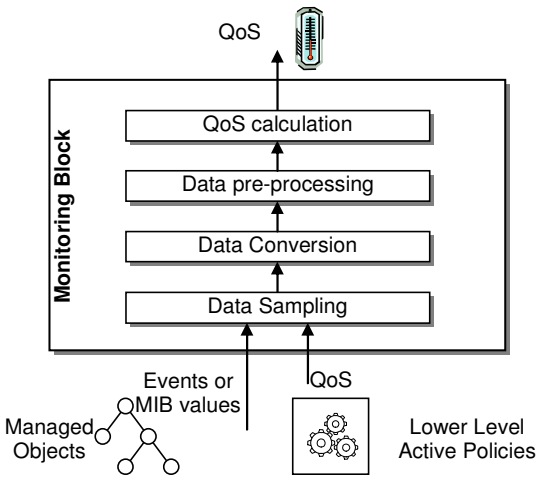

Fig. 2. Active policies' monitoring module

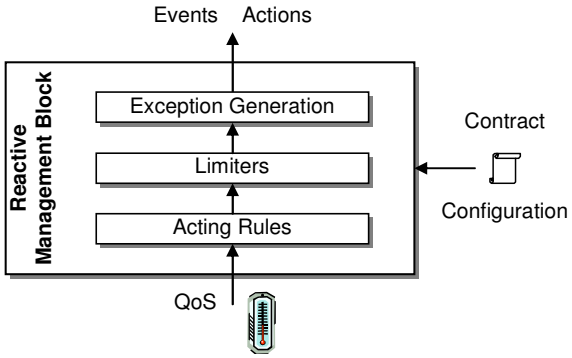

Fig. 3. Active policies' reactive module

Acting rules can have a set of thresholds, with or without hysteresis, count events, act based on time, use feedback control, or combinations of these. The rules may be used in different active policies, just by configuring the parameters they use and modify.

Basically, active policies act over the network, when certain thresholds on the QoS are crossed, to avoid SLA violation. Naturally, policies have to start acting long before the QoS limits are crossed. The actions are usually stronger as the limits are approached. On the other hand, if the QoS is better than specified in the SLA, the reverse actions are possible to reduce the service provider costs, without SLA viola-

tion. For instance, an admission control system may admit some more flows, allowing the service provider to charge more. To ensure stability, the programmer has to know if a certain action contributes to improve or worsen the QoS and the actions in the direction of improving the QoS should be more intense than the actions in the direction of degrading the QoS. If the contribution of a certain action is unknown, a perturbation method [6] may be applied to determine the contribution. The idea in this method is to force explicit system perturbations and use a statistical model to estimate dependency strengths for each one.

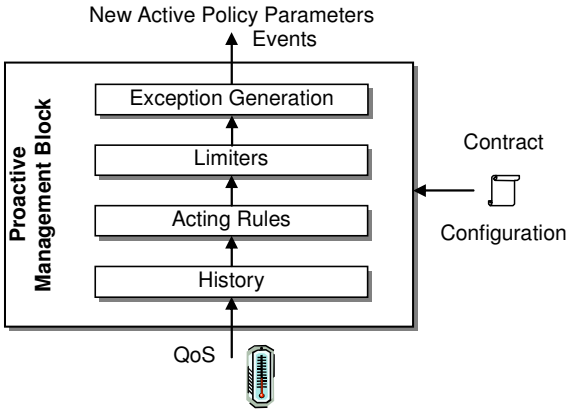

Fig. 4. Active policies' proactive module

Examples of actions the active policies might perform are: modifying Random Early Detection (RED) parameters for increasing packet discard, especially for out-of-profile packets; restricting the admission of new flows into the system; activating extra lines or increasing the bandwidth available for some paths; and selectively degrading the class of certain flows.

The active policies architecture should be deployed using a management by delegation technique [7], placing the active policies near the objects they manage. The delegation adds flexibility as it allows system modifications during operation. The higher level policies have a global vision of the network state, while the lower level policies are near the managed equipment, minimizing management and signaling traffic. This architecture adds flexibility, increases scalability and fault tolerance.

3 System Architecture

3.1 Active Policies Hierarchy

A scenario of a DiffServ network was built. The service provider wants to offer some QoS to users. The DiffServ model allows offering scalable QoS differentiation by

aggregating the traffic in three traffic classes: Expedited Forwarding (EF), Assured Forwarding (AF) and Best Effort (BE). Packets are classified and marked on entering the network, being forwarded to their destination according to the per-hop behavior of their class. The EF class emulates a virtual circuit. The AF class offers some guarantees and the BE provides no guarantees at all.

Figure 5 shows the active policies deployed and how they are related to each other. The different levels in figure 5 correspond to different abstraction levels. The top-level QoS requirements are refined through each abstraction level. For each abstraction level, QoS parameters that can be measured are identified, the corresponding QoS requirements are determined and actions adequate for that abstraction level are specified. The active policies cooperate with each other to fulfill the QoS objectives for each level and consequently, the top-level QoS objectives.

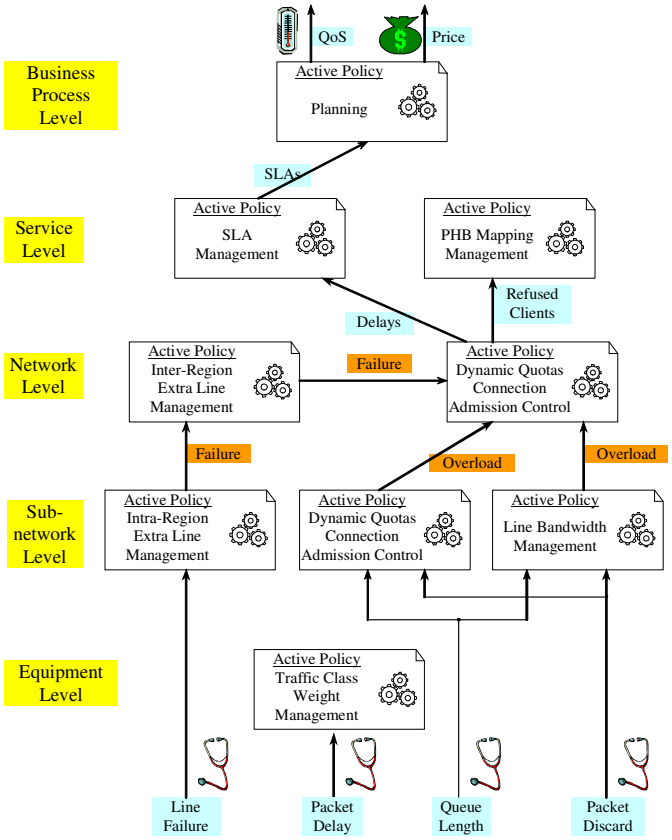

Fig. 5. Active policies used

At the Equipment level, policies are local to each device. Policies at this level monitor the delay in each traffic class and dynamically adjust the weight of the different classes and out-of-profile packet discard. In this way, the existing network re-

sources are divided through the traffic classes to ensure the QoS relations between classes, no matter the load of the classes.

The Network level is sub-divided in two sub-levels to ease implementation and improve scalability. Sub-network level policies are responsible for a region, or a specific equipment. Network level policies are responsible for the entire network. Policies at this level perform measurement based admission control, select backup lines and dynamically adjust the bandwidth available for some paths.

The Service level is the first that sees end-to-end QoS parameters. Policies at this level monitor end-to-end QoS and try to enforce end-to-end SLAs by adjusting parameters of lower level policies that do not have this end-to-end view. Examples are: increasing out-of-profile packet discard, reducing flow admissions, modifying the bandwidth for some paths. Policies also selectively degrade the traffic class of some flows when there is overload.

A more detailed description of the operation and configuration of each active policy used is given in [4].

As is it difficult to specify the exact thresholds where the policies should operate, a planning policy, at business process level, makes long-term adjustments to ensure a more precise SLA fulfillment.

3.2 SLA Management without Planning

First, the results without the planning policy are presented and analyzed. The results with planning will be analyzed in the next subsection.

The results presented were obtained with the NS2 network simulator [8], for a network with 15 core nodes. The simulations were made with the exact same traffic pattern in two different situations: without active policies and with the active policies operating. Several simulations were made with different random number seeds and the 95% confidence interval was plotted in all the graphs.

Figure 6 shows the average end-to-end delays obtained with the following QoS objectives: *SLA(delayEF)* = 60 ms and *SLA(delayAF)* = 120 ms, and all policies operating, except the planning policy. These results show it is possible to enforce these SLAs with some error. The error from the EF SLA objective is about 24 to 32% without the policies and about 7 to 16% with the policies. The error from the AF SLA objective is about 12 to 26% without the policies and about 13 to 24% with the policies. Although the results are better with the active policies, they show it is difficult to set the exact policies operation points to have precise SLA enforcement.

Figure 7 shows the average traffic load in each class for each border node (PoP, or Point of Presence), where traffic is generated. Without the active policies, a fixed worst case admission control is used. Under these conditions, the network is fully loaded with 15 users connected per node, starting to refuse new users above this limit. On the other hand, the active policies use a measurement based admission control that can allow more connections in the AF class. Indeed, as the traffic in many flows is bursty, the average load in the network is much lower than the worst case load for a given user connection pattern. Modifications in the bandwidth division for each class also allow a slight increase in the number of EF users. The decrease observed in the number of EF clients for a load of 20-50 connections, when the active policies are

used, is due to the downgrade of the traffic class of some flows by the mapping management active policy, as the network load increases. The use of the active policy has the advantage that, instead of having an indiscriminate downgrade of the class of flows, only selected flows have their class downgraded according to the existing load. From the graph of the average BE clients, it can be seen that there are less BE users when the active policies are used. This is caused by less flows having their traffic class downgraded when the active policies are used and results in increased service availability and a profit increase for the service provider.

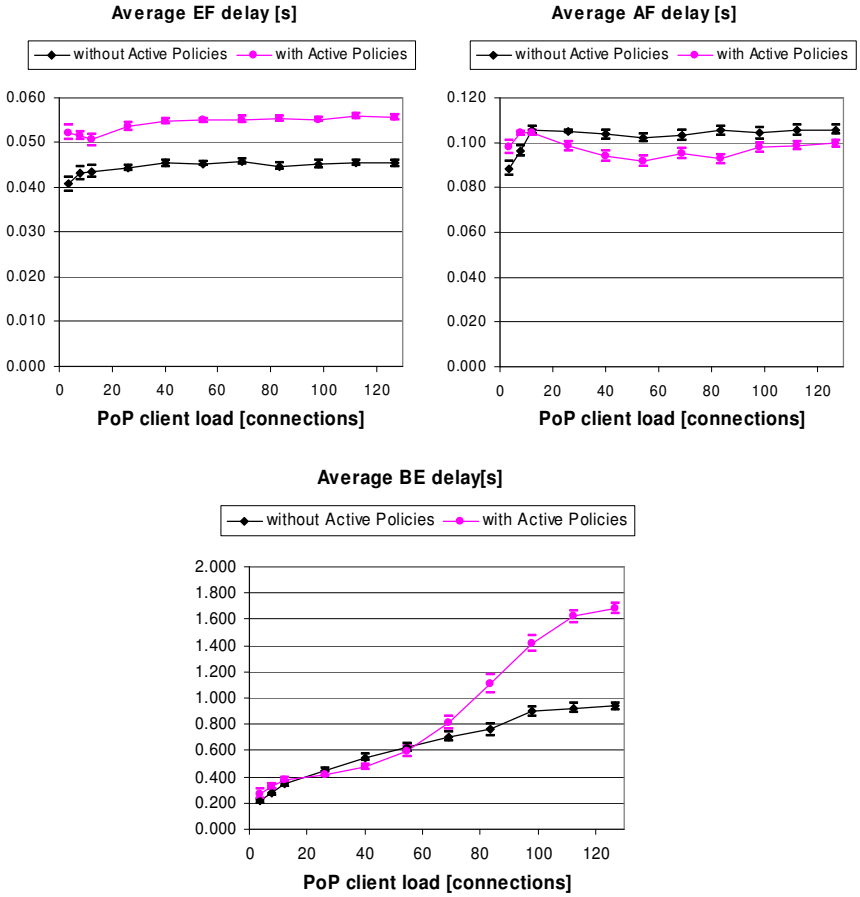

Fig. 6. Average delay per traffic class

Figure 8 shows the average throughput for the AF traffic class. The throughput decreases when the active policies are used. There was no packet loss both for EF and in-profile AF traffic. This was one of the high-level QoS objectives for these classes. Having no packet loss for the AF class means that the decrease in AF throughput is

caused by less out-of-profile traffic being transported. Since more flows are transported, and the throughput specified in the contract is assured, the service provider profit increases.

Figure 9 shows the average jitter for the EF and AF classes. The active policies cause an increased jitter, as they reconfigure the network frequently. Each modification in the network changes the load pattern and consequently changes the end to end delay, causing an increased jitter. This is especially true when the delay is below the limit imposed by the SLA and the bandwidth for a path is reduced, or the admission of additional users increases the traffic load. However, in the EF class, there is a reduction of the jitter for a certain load range, as some bursty flows have their traffic class downgraded to a lower priority class for these loads. Globally, the disadvantage of an increased jitter is compensated by the improvement in other QoS parameters.

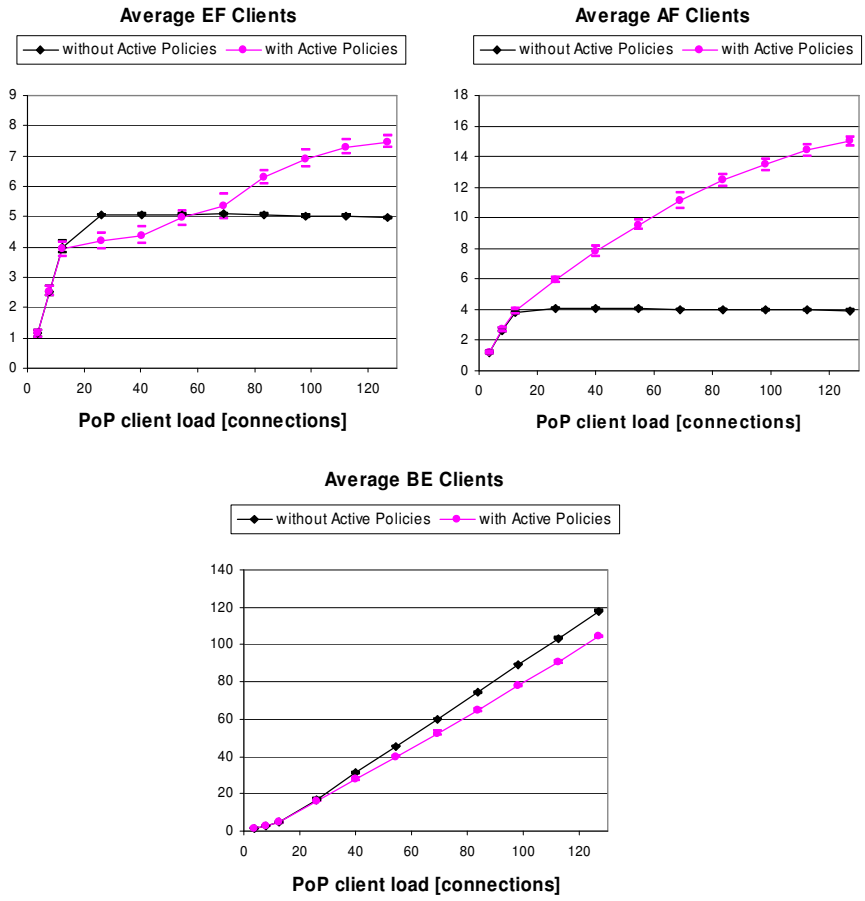

Fig. 7. Average traffic per traffic class

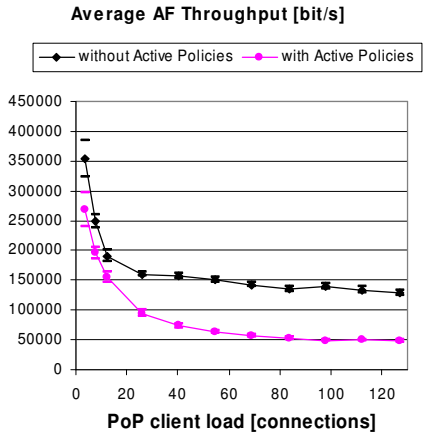

Fig. 8. Average throughput for the AF traffic class

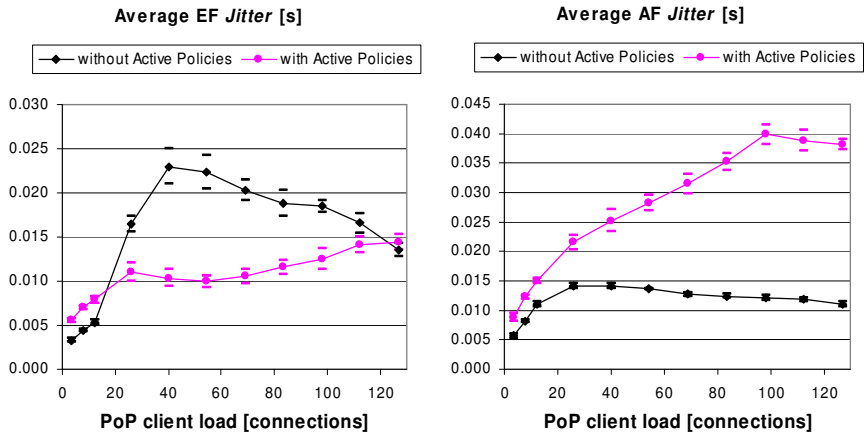

Fig. 9. Average jitter per traffic class

3.3 Planning Policy

The top level planning active policy adjusts the SLAs of the SLA management active policy to obtain better long-term results.

Now, the previous delay objectives, as required in the SLA, are used to set the *SLA(delayEF)* and *SLA(delayAF)* to be used by the SLA management active policy. Both are modified by an integral feedback system as shown in figure 10. The integra-

tor gain is set to 0.25, for both classes, to make slight changes in the SLAs for each sampling period. The sampling period is large as compared to the network delays, but much smaller than the SLA verification period. The planning policy works at higher abstraction level than the other active policies, so its timescale of operation is much larger than the timescales of lower level policies. An anti-windup mechanism was added to the integrators to limit the SLA values to a range of 0.25 to 4 times the required SLA.

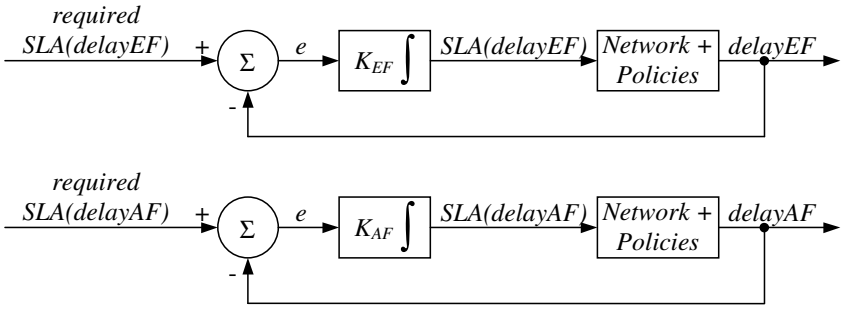

Fig. 10. SLA control system

Figure 11 shows the average end-to-end delays obtained with all the active policies, including the planning policy, as compared with those without any policies. These graphics show that the static error from the SLA objective is greatly reduced by the planning policy to about 1 to 8% for the EF class and 1 to 6% for the AF class. Now the delays are about one order of magnitude nearer the specified value of 60 ms for the EF class, and 120 ms for AF.

As for the other QoS parameters, such as the jitter, the throughput, the packet loss, the service availability (number of users accepted), there were no significant differences when the planning policy was used.

The objectives of 60 ms for the delay in the EF class and 120 ms for the AF are achievable. Naturally, not all SLA objectives are possible to enforce. But even when it is not possible to enforce the objectives, the active policies still work properly and do their best to place the QoS as near as possible to the objectives. The system is stable even in traffic overload conditions.

4 Related Work

The works of [9][10] present policy hierarchies for distributed systems. However, these works do not integrate with service level management, nor refine the requirements through several policy abstraction levels that act over the network down to the equipment, nor allow easy policy reuse. The policies of [11] restrict the network behavior, instead of improving its operation as in the case of the active policies. The work in [12] focuses on the policy language and provides some examples of a differ-

entiated services network with SLAs. However, the system lacks an automatic tuning mechanism and does not cover end-to-end QoS.

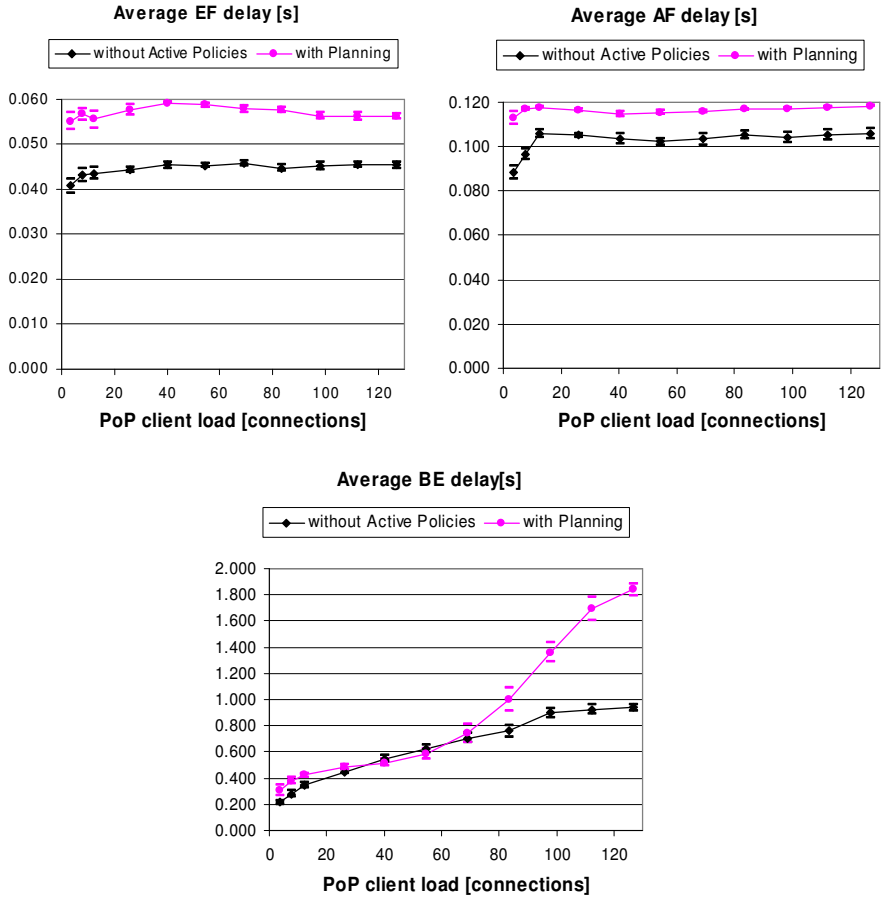

Fig. 11. Average delay per traffic class

The work in [13] presents an architecture for QoS management integrated with traffic engineering. Although there are some similar aspects to our work, the active policies hierarchy has the advantage of providing several complementary mechanisms to enforce QoS and a planning policy to improve accuracy.

The work in [14] uses joint buffer management and scheduling to provide proportional and absolute QoS guarantees. However, this work does not have several abstraction levels and does not cover end-to-end QoS.

5 Conclusion

It can be concluded that the QoS specified in SLAs can be provided within certain limits by using active policies that dynamically adjust network parameters to optimize network performance.

The planning policy performs automatic threshold adjustments to provide a more precise fulfillment of QoS requirements, circumventing the difficulty in setting policy operating points. However, it is difficult to evaluate to which limits the SLAs can be assured. This is a difficult topic that may be addressed in future research.

A set of acting rules can be used to build the majority of the active policies. The acting rules can be reused for different situations, just by changing the system variables they monitor, the system variables they control and the configuration parameters of the rules.

The active policies hierarchy provides several abstraction levels, with greater flexibility, autonomy, improved scalability and fault tolerance, as compared with non-hierarchical alternatives.

The architecture presented assumed a single service provider. The expansion for multiple service providers is a possible future research direction. The use of a price model [15] is a further possible future research direction.

References

1. Terms and Definitions Related to Quality of Service and Network Performance Including Dependability. ITU-T Recommendation E.800. August 1994
2. A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, S. Waldbusser. Terminology for Policy-Based Management. IETF RFC 3198. November 2001.
3. Lundy Lewis. Service Level Management of Enterprise Networks. Artech House. September 1999. ISBN: 1580530168.
4. Paulo Pereira, Djamel Sadok, Paulo Pinto: Service Level Management of Differentiated Services Networks with Active Policies. Proceedings of the 3rd Conference on Telecommunications (ConfTele'2001), Figueira da Foz, Portugal, April 2001, pp. 542-546. ISBN: 972-98115-2-0.
5. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss. An Architecture for Differentiated Services, IETF RFC 2475, December 1998.
6. A. Brown, G. Kar, A. Keller. An Active Approach to Characterizing Dynamic Dependencies for Problem Determination in a Distributed Application Environment. Proceedings of the 7th International IFIP/IEEE Symposium on Integrated Management (IM'2001), Seattle, EUA, 14-18 May 2001.
7. Germán Goldszmidt, Yechiam Yemini. Distributed Management by Delegation. Proceedings of the 15th International Conference on Distributed Computing Systems, June 1995.
8. UCB/LBNL/VINT Network Simulator (version 2). http://www.isi.edu/nsnam/ns/
9. René Wies. Policies in Network and Systems Management - Formal Definition and Architecture. Journal of Network and Systems Management, Plenum Publishing Corp., **2**(1):63-83, March 1994.
10. Thomas Koch, Christoph Krell, Bernd Krämer. Policy Definition Language for Automated Management of Distributed Systems. Proceedings of the 2nd International Workshop on Systems Management, IEEE Computer Society, 19-21 June 1996, Canada.

11. M. Sloman, J. Magee, K. Twidle, J. Kramer. An Architecture for Managing Distributed Systems. Proceedings of the 4th IEEE Workshop on Future Trends of Distributed Computing Systems, Lisbon, Portugal, IEEE Computer Society Press, pp. 40-46, 22-24 September 1993.
12. L. Lymberopoulos, E. Lupu, M. Sloman. An Adaptive Policy Based Framework for Network Services Management. Plenum Press Journal of Network and Systems Management, Special Issue on Policy Based Management, **11**(3):277-303, Sep. 2003.
13. P. Trimintzios, G. Pavlou, P. Flegkas, P. Georgatsos, A. Asgari, E. Mykoniati. Service-Driven Traffic Engineering for Intradomain Quality of Service Management. IEEE Network, **17**(3):29-36, May/June 2003.
14. Nicolas Christin, Jörg Liebeherr. A QoS Architecture for Quantitative Service Differentiation. IEEE Communications Magazine, **41**(6):38-45, June 2003.
15. Luiz A. DaSilva. Pricing for QoS-Enabled Networks: A Survey. IEEE Communications Surveys, **3**(2):2-8, 2nd Quarter 2000.