Quality of Service Management in IP Networks using Mobile Agent Technology

Telma Mota[1] , Stylianos Gouveris[2] , George Pavlou[2] , Angelos Michalas[3] , John Psoroulas[3]

1 PT Inovação, Aveiro, Portugal
telma@ptinovacao.pt
2 Centre for Communication Systems Research, School of Electronics, Computing and Mathematics, University of Surrey, Guildford, Surrey, GU2 7XH, UK
{S.Gouveris, G.Pavlou}@eim.surrey.ac.uk
3 National Technical University of Athens, Division of Computer Science, Department of Electrical and Computer Engineering, GR-15773 Athens, Greece
michalas@acadinfo.central.ntua.gr

Abstract. Telecommunication network infrastructures become more and more distributed and heterogeneous, with various network technologies integrated (e.g. ATM, IP, ADSL), and several types of equipment coexisting, typically from different vendors. At the same time, services are becoming more and more demanding in terms of quality of service. In this context, efficient network management is of paramount importance for network operators. The telecommunication business does not any more depend on the type and cost of the services but mostly on their quality and how quickly they can be introduced. In this context, management platforms should be flexible and intelligent enough to dynamically configure, monitor and eventually reconfigure the network. On one hand, these platforms should be able to inform the clients about the quality of the requested services and on the other hand, network administrators must be able to manage their network resources in order to efficiently provide quality of service. Service Level Agreements (SLAs) between network operators / service providers and clients should be honored in a manner absolutely transparent to the users. This paper describes the network management architecture followed by the European IST (Information Society Technology) Project MANTRIP (MANagement, Testing and Reconfiguration of IP based networks using Mobile Software Agents) [1]. This architecture is based on both standard and tailored APIs. The prototype developed during the MANTRIP project is described in detail. Its main objective is to manage the Quality of service in IP Networks, with Mobile Agent Technology (MAT) being used to implement the specified APIs in the QoS management system and IP Differentiated Services (DiffServ) being used as a test case.

1. Introduction and Background

Traditionally, connection-oriented network technologies (e.g. PDH, SDH,ATM) have been the main focus of network management. Due to the recent use of IP backbones and emerging IP technologies that support QoS (e.g. IntServ, DiffServ, MPLS), the network management area has been enlarged in scope. The use of IP technologies is growing; all-IP environments covering from access to core-networks are increasing in the telecommunications world. The problem consists in dimensioning and correctly administering network resources according to the new service requirements.

IP network technology enters the telecommunication market due to both its simplicity and low price. It can be suitable for supporting various types of applications but it is not suitable for services with high quality of service (e.g. real time audio and video services). The new IP technologies with QoS capabilities try to support these requirements but at the same time pose a new problem: effective network resource management. IP routers must be managed, i.e. configured, monitored and dynamically reconfigured in order to meet user requests. Clients are becoming more and more demanding, services must be provided fast and executed efficiently. Therefore, operators and service providers should be prepared to quickly respond to client requests. Robust and flexible management platforms are key enabling factors to help providers in honoring their contracts (i.e. Service Level Agreements).

Existing network management systems are typically dedicated to management of certain type of network resources. These are often vendor and technology specific and make use of centralized architectures, which are not flexible in principle. Management information can grow significantly and introduce substantial delays in the network. Scalability of network management systems is frequently a limitation, which determines their applicability in managing large distributed networks, such as IP-based Next Generation Networks (NGNs).

Recently, Mobile Agent Technology (MAT) [5], [6], [7], [8] has emerged as a promising solution towards implementing strategies that distribute and automate management tasks. Even though there are still several open research issues, mobile agent technology appears mature enough to support distributed and decentralized network management. Mobile Agents can easily migrate to remote locations, execute their tasks and come back with results. This capability allows agents to travel from one node/machine to another and perform a repetitive task or, when appropriate, several instances of the same agent can be created in order to perform the same task in parallel.

Standardization organizations made the effort of finding solutions for well-known problems such as security, portability, mobility, resource management and discovery. As a result, several mobile agent platforms are already available and new ones will become available in the future.

The objective of the MANTRIP project is the application of this new distributed and autonomous software technology to the management of IP networks. Experiences from the use of this emerging paradigm and technology in the context of MANTRIP are presented in this paper, which has the following structure. Section 2 presents a brief overview of MANTRIP system. Section 3 is the key section of this paper and

presents in detail the agent-based MANTRIP QoS Management application. Finally, Section 4 presents our experiences with agent technology in the context of QoS management in IP networks.

2. The MANTRIP project

The MANTRIP (MANagement, Testing and Reconfiguration of IP based networks using Mobile Software Agents) project exploits the mobile agent features in order to provide novel management systems and applications [2]. Each of the developed applications integrates a set of software modules in a common environment – a Mobile Agent Platform, either Voyager [34] or Grasshopper [33]. Using different applications, both the network operator/administrator and users are able to access management information on end-to-end performance, alarms, protocol conformance, QoS, etc. and interact with the network in order to configure and reconfigure it when required.

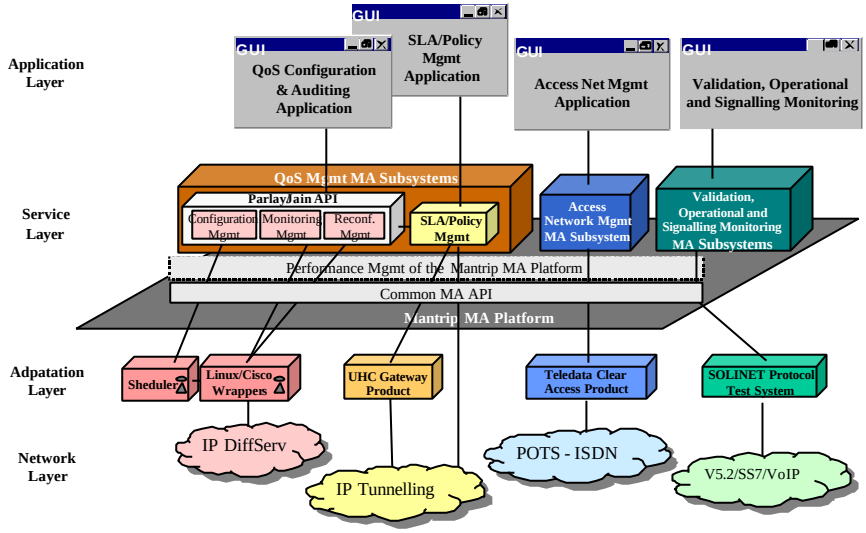

Fig. 1. MANTRIP Management System

The overall architecture is depicted in Figure 1. The different software modules that constitute the overall management system are organized in different layers of concern.

- **Application Layer** includes both existing management applications and applications developed within the MANTRIP project. They are:
  - **Conformance Testing and Signaling Monitoring Application** [27].

- **Configuration and Alarm Management application for Access Networks Application** [28].
- **SLA Management Application and Policy Network Based Management** [29].
- **QoS Management Application** that is the main focus of this paper. It allows dynamic configuration, monitoring and reconfiguration of QoS parameters in IP DiffServ Network resources [32] [30] [31].

- **Service Layer** contains the software modules – services that support the execution of various applications. These are services like the QoS Configuration and QoS Monitoring as well as services enriching the MA platform's capabilities. As an example, the Common MA API abstracts away from the type of MA platform. Either Grasshopper or Voyager can be used without having any impact on the other services.
- **Adaptation Layer** is responsible for hiding the protocol details from the service layer. It includes the network adapters and wrappers.
- **Network Layer** includes the network resources and existing network management platforms to be managed by the mobile agent platform.

This layered architecture makes the overall MANTRIP management system more flexible.

On the other hand, Mobile Agents can be used to decentralize the processing of some of the most important management tasks. Other technologies may be used for this purpose but the most common paradigms like client/server, may easily lead to poor performance when extensively used in management systems due to the amount of information that needs to be exchanged between central and remote nodes. Decentralization improves scalability of management systems. When networks of considerable size need to be managed this is defiantly an important feature. Moreover, agents can easily be replaced in real time. Heavy management tasks as network configuration, performance monitoring and alarm processing can be performed by mobile agents in a remote fashion. When a new task needs to be accomplished, a new agent can be created and migrate to a remote place to perform its task without disrupting the normal functioning of the local system. The need of transferring big amounts of information to central stations can drastically decrease by using mobile agents.

In summary, agent technology, when properly used, can become a powerful tool to increase software flexibility, distribution and decentralization.

3. QoS Management Application

The QoS management architecture has been defined based on the following main requirements:

- ***Protocol and Vendor Independenc***e: Introducing new types of routers, (e.g. *Cisc*o, *Linu*x, *Redstone)*, eventually coming from different vendors, should not have impact on the applications.

- ***Network Technology Independence***: The impact of introducing new IP-based technologies (DiffServ, MPLS…) should be minimized.
- ***Openness and Flexibility***: The interfaces between the different layers should be open, flexible, and standard whenever possible in order to promote quick development of management applications.

Based on these requirements, the MANTRIP project built an architecture that combines open and standard APIs with mobile agent technology. It tried to take advantage of both worlds. Figure 2 shows the QoS Management Architecture. In between the software modules represented below, standard and tailored APIs have been used.

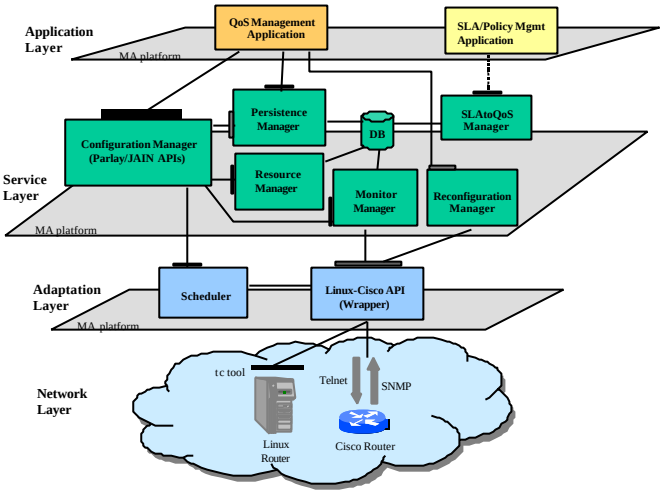

Fig. 2. QoS Management Architecture

A. QoS Configuration and Monitoring

The QoS Management Architecture includes the following components and subsystems:

- *The **QoS Configuration and Monitoring Application*** offers a Graphical User Interface (GUI) to both network administrators and users. By using this GUI, administrators may profit from an extensive set of privileges: change dynamically the network configuration QoS parameters (e.g. max BW available per class of service, queue size, buffer size), monitor QoS of the traffic produced by users (delay, jitter, loss and bandwidth), change monitoring periods, create/delete users, access points, routers, create/delete and modify QoS templates, etc. The user can have access to all the information available in the GUI but can s/he can not change it, s/he can only request uni or bi-directional resource reservation (i.e. set-up a scheduled VPrP – virtual provisioned pipe) by choosing a QoS template and certain value for the

bandwidth, minor or equal to the maximum bandwidth available per class of service. The QoS templates made available to users may be constrained by the SLA defined externally by another application shown in the picture above, such as the SLA/Policy Management application.

- The **Connectivity Management Subsystem** implements in Java the Parlay Connectivity Manager API [16], an open and standard Application Programming Interface (API) specified by the Parlay Group [15]. The implementation of this API includes a set of service sub-modules (Connectivity Manager, Persistence Manager and Resource Manager) that offer layer generic IP connectivity management capabilities to the application. By using this subsystem, administrators can manage the network and populate the database with both application and network specific information while users can request/schedule resources and QoS. The Connectivity Management subsystem implements the mobile agents that are sent as close as possible to the routers to perform the requested connectivity tasks.
- The **Performance Monitor Management Subsystem** offers to both users and administrator the capability of monitoring the QoS parameters of the configured connections. The Monitoring subsystem makes use of Active and Passive monitoring techniques for obtaining delay/jitter and bandwidth utilisation/loss statistics respectively. The user of this system has also the capability of modifying the granularity and report period of either monitoring service on-the-fly. This module implements the mobile agents that are sent as close as possible to the routers to perform monitoring tasks.
- The **Configuration and Reconfiguration Management Subsystem** allows the administrator to (re-)configure certain QoS parameters in routers. On one hand, it caters for the initial configuration of the routers (i.e. define parameters for the DiffServ classes of service) and on the other hand, if a certain path flow is violating the thresholds defined by the QoS template, the administrator may trace the route of a certain VPrP (if static routing is being used), get the queue load of the routers involved and reconfigure them in order to improve the end-to-end QoS. Through this subsystem it is possible to configure and reconfigure the Random Early Detection (RED) parameters, the queue size per class of service and the maximum bandwidth allocated to each class. Furthermore, the reconfiguration module estimates new bandwidth values according to both the queue load per class, and the required delay spacing among classes defined by the user. This module implements the mobile agents that are sent as close as possible to the routers to perform (re-)configuration tasks.
- **Scheduler** and **Wrappers** - include the static agents that are located as close as possible to the routers in order to serve specific requests from visiting agents, which cannot access directly the network layer. A common tailored API (referred in Figure 2 as the Linux-Cisco API) has been defined to configure and obtain QoS monitoring information (e.g. loss and bandwidth) from both Linux and Cisco routers. Due to the limited information that is possible to gather from the routers, some mobile agents are used to “emulate” traffic and measure some additional parameters (e.g. delay and jitter) in the context of Active Monitoring. Linux and Cisco wrappers implement this

common API and hide from the upper layer the type of router being used in the network and the protocol details. The scheduler is located only in the edge routers and keeps track on the reservations and the resources available.

B. JAIN/Parlay API

A number of interfaces were considered for adoption and inclusion in the MANTRIP system, including the interfaces under development from: TSAS (OMG Telecommunications Domain Task Force Activities), OSA API (part of the 3GPP standardization work), JAIN (Java API Initiative), Parlay API, Multi-Service Forum. Due to the functionality and status of the specification, the level of openness and the likely acceptance by the industry, the Parlay interfaces were adopted as the basis of inclusion in the MANTRIP system. A number of extensions, adaptations and clarification were made.

The Parlay Group [15] is an open multi-vendor consortium formed to develop open technology-independent Application Programming Interfaces (APIs) enabling third parties to develop applications and technology solutions to operate across multiple networking platform environments. Most of the Parlay specifications have been submitted and accepted by standardization organizations such as 3GPP and ETSI. Key operators and vendors such as BT, Telecom Italia, France Telecom, IBM, SUN, Nokia, Ericsson, Lucent, Alcatel, Siemens, Nortel and Cisco are members of the Parlay consortium.

The MANTRIP project adopted the Parlay Connectivity Manager API [16] in order to configure QoS parameters in IP DiffServ Networks. This is an open API that may be used by any application developer (users, service providers, network operators, brokers, etc.)

Besides following the Parlay specification the project also decided to follow the approach suggested by the JAIN initiative [17]. This group is supported by Sun Microsystems and has a similar objective to Parlay but more focused in scope, that is to define a set of open APIs in Java that abstract from specific network protocols and ease the fast development of next generation applications over a Java platform. As in JAIN, IP network management APIs have not yet been defined, so we created two levels of Java APIs. One, at the network layer - the Linux-Cisco API – that abstracts from specific type of routers and protocols and another one, at the service layer - a Java implementation of the Parlay API – that offers to applications common access to network capabilities. A Java Parlay API may be used for local access while a Parlay CORBA API is most recommended for application remote access.

C. Agent Interaction – Network Resources

In our prototype the agents have been placed as shown in Figure 3:

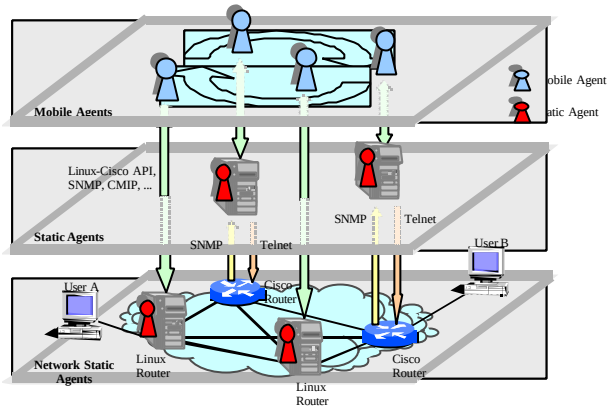

Fig. 3. Agent Interaction

In our **network** we have both Linux and Cisco routers that constitute the IP DiffServ environment.

The **static agents** are responsible for controlling directly the routers. They are called static because they migrate once to the routers, or as close as possible to them, and they stay there until new functionalities need to be provided and they need to be replaced. These agents interact with the network elements in order to perform the tasks the mobile agents tell them to do. They offer the Linux-Cisco API towards the mobile agents that will be using their services to configure, monitor and reconfigure QoS parameters in the DiffServ routers.

The *wrappers* are static agents, which offer the following set of interfaces:

- ***Edge_ConfiguringSLS*:**

This interface allows for edge routers configuration, i.e. it allows for configuring the classifiers and meters according to a SLS. All entries in the classification table can be retrieved. Overlapping filters are allowed, provided that they have different priorities. However, it is not the rule of the API implementation module to verify overlapping of filters.

- ***Monitoring*:**

This interface allows for monitoring the amount of traffic forwarded and dropped by each class. It should be used to get the so-called Passive Monitoring Measurements, i.e. Packet Loss and Throughput.

- ***Core_Configuring*:**

This interface is invoked by the administrator in order to provide basic configuration to all routers, Core and Edge.

The *Scheduler* is also a static agent used to manage the time that *connectivity* should be active according to the user requests (or SLA). It contains one interface:

- ***IScheduler***

The **mobile agents** are created by the service subsystems and migrate to places where the static agents reside.

D. IP DiffServ Network Configuration

As mentioned before, the QoS management prototype operates on an IP DiffServ network. By using the Linux-Cisco API it is possible to configure the internal modules of each DiffServ node, as represented in figure 4. The filters of a router are composed by two modules: a *classifier* that identifies the traffic flow and a *meter* that characterizes the flow in terms of bandwidth. The meter supports TCM marking *(Three Color Marker – Green, Yellow, Red*) [18],[19]. The *marker* is responsible for effectively marking the IP packets by assigning a certain *DiffServ Code Point* (DSCP) to each packet according to its QoS characteristics. The DSCP is just a binary code, which is placed in the Type of Service (ToS) field of the IP header. The Per Hop Behaviors (PHBs) are defined by both the *shaper/dropper* and the *scheduler*. By configuring these modules, we define how each DSCP is used by routers to select which packets should be forwarded first to the next router. Besides the traditional *Best-Effort*, there are two other PHBs: *Expedited Forwarding* (EF) [21] and *Assured Forwarding* (A*F)* [20]. These behaviors differ from each other in terms of QoS guarantees. Distinct configuration of the waiting queues must reflect this difference.

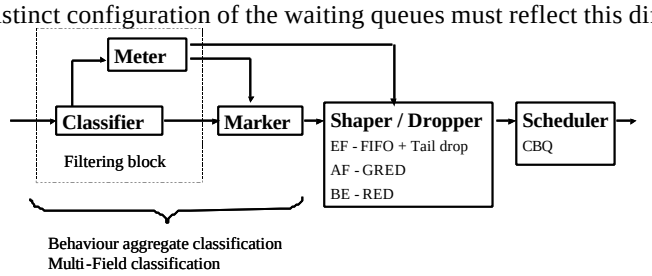

Fig. 4. IP DiffServ Node

In order to implement the EF PHB, where the objective is to ensure a class of service with a minimum guaranteed bandwidth (configurable) and low losses, jitter and delay, the FIFO *(First In, First Out*) queuing mechanism has been used; packets are delivered in the same order in which they arrive to the queue.

The PHB AF defines four classes of service, each of which contains three subclasses (AF11, AF12, AF13, AF21, AF22, AF23, AF31, AF32, AF33, AF41, AF42, AF43). The difference between them consists on the priority given to the packets that must be dropped. The AF DiffServ PHBs have been implemented using the *Random Early Detection* (RED) queuing mechanisms, where packets are dropped according to both queue occupancy and congestion probability.

Traffic from all the queues flows into the module that assigns the packets to be sent *(router scheduler*). This module was implemented by using *Class Based Queuing* (CBQ).

E. The QoS Management Application – Use Cases

The Graphical User Interface of the QoS management application is depicted in the following figure.

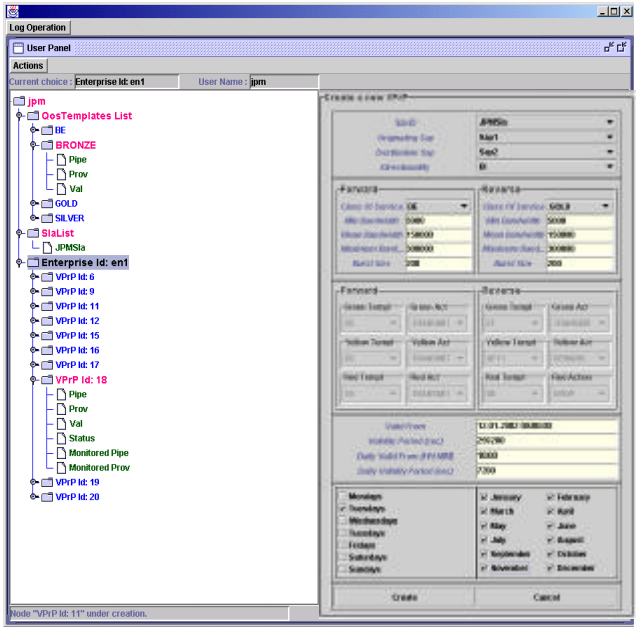

Fig. 5. QoS Management Application – Configuration GUI

- Configuration Scenario

By using this GUI the user can make connectivity reservations by choosing values for the following set of parameters:

- Class of Service *(Bronze, Gold, Silver, Best-Effort,* or others – Depends on QoS templates available)
- Service Access Points (SAPs) origin and destination
- Directionality
- Minimum guaranteed bandwidth
- Time scheduling

Taking the example presented in Figure 5, the user "jpm" is interested in using the network resources, every Tuesday between 10 pm and 12 pm, starting from 2002 December 12. Since s/he would like for example to watch a movie, s/he is interested in a good quality of service in one direction but not so good in the opposite direction. Therefore s/he requests a bi-directional VPrP between SAP1 and SAP2 and chooses BE in the forward direction and Gold in the opposite direction. Furthermore s/he knows that the movie will not need much more than 2Mb/s. So, s/he will request to use this bandwidth that is a percentage of the total one available for the Gold class of service. This means that the traffic bellow 2Mb/s, i.e. the well-behaved traffic will be marked as Green and will have the highest priority (mapped into the EF DiffServ class of service). The traffic, which exceeds this value in 10% will be marked as Yellow (mapped into one of the AF DiffServ classes of service). If it exceeds more than 10% it will marked as Red (BE) and will have the lowest priority. When the day comes the routers will be configured according to the user requirements. During the activation time the user will be able to monitor QoS of the traffic being sent. S/he will

also be able to see if the contract is being violated with respect to the QoS requirements.

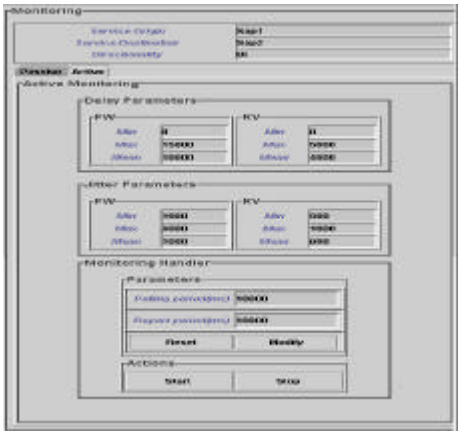

Fig. 6. QoS Management Application – Monitoring GUI

- *Monitoring Scenario*:

Through the monitoring interface (Figure 6) the user may ask to monitor the VPrP and observe how the QoS parameter values change. Performance monitoring tasks involve both active and passive measurements. Active measurements are taken by inserting (low-impact) test streams (delay, jitter, etc.), while passive measurements are taken by analyzing the traffic passing from a network element (used bandwidth, loss, etc.). The system provides performance reports and notifications in a scheduled and 'on-the-fly' manner, respectively.

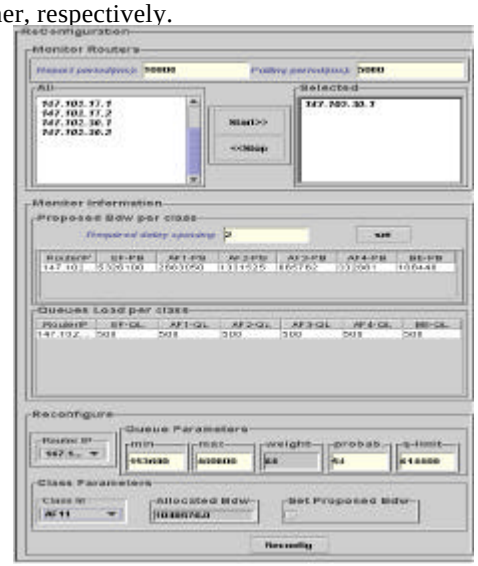

Fig. 7. QoS Management Application – Reconfiguration GUI

- *Reconfiguration Scenari*o:

An administrator can log into the application and get all the router interfaces involved in a VPrP by using the trace_route functionality. He/she can enter the reconfiguration screen, as depicted in Figure 7, select one or some of the traced IP addresses and get the RED parameters, the queue size and the actual queue load per class of service. The user can then reconfigure the router RED parameters and the queue size per class of service. Optionally, s/he can select the proposed bandwidth value per class of service given by the management system. After reconfiguring the administrator can go back to the monitoring GUI to check if the QoS parameters improved and, specifically, if the required delay spacing among the reconfigured classes is achieved.

4. Conclusions

In this paper we described the architecture, design and API issues for a management system based on mobile agent technology that supports quality of service management in IP networks. Using mobile agents as the basic design andimplementation technology for our system helped us to decentralize configuration and monitoring tasks. In addition, it promoted good software design and made relatively easy the implementation of such a complex system. On the other hand, the mobile agent platforms we used were somewhat immature and not yet streamlined for efficiency. Despite that, the savings for complex configuration and monitoring tasks are significant in comparison to client/server protocol-based or distributed object technologies; we are planning to quantify that in future papers. But most important, mobile agent technology supported the "programmability" of network nodes for configuration and monitoring purposes in a rapid manner, assuming only raw management capabilities available. We are confident that mobile agent technology will play a significant role for decentralization and programmability in the next generation of management systems supporting multi-service IP-based networks.

5. Acknowledgements

The authors would also like to acknowledge the colleagues in the MANTRIP project who did undertake design and implementation work related to the overall MANTRIP Management System.

References

[1] Web Site MANTRIP http://www.solinet.com/mantrip
[2] Deliverable D21 " System Requirements and High Level Architecture", Project MANTRIP, December 2000.

[3] Baker, F., Chan, K., Smith, A., " Management Information Base for the Differentiated Services Architecture", http://www.ietf.org/internet-drafts/draft-ietf-diffserv-mib-06.txt, Internet Draft, November 2000
[4] Bernet, Y., et al, " A Framework for Differentiated Services", draft-ietf-diffserv-framework-0.2.txt
[5] Bieszczad, A., Pagurek B., White, T., " Mobile Agents for Network Managing", 1998
[6] Caripe, W., Cybenko G., Moizumi K., Gray R., " Network Awareness and Mobile Agent Systems", IEEE Comm. Mag., July 1998, pp.44-49
[7] Dale, J., " A Mobile Agent Architecture to Support Distributed Resource Information Management", 1996
[8] Galis, A., Rao, S., " Application of Agent Technology to Telecommunication Management Services," in "On The Way To Information Society," ed. Magedanz et al, pp. 409-415, IOS Press, Amsterdam, The Netherlands, ISBN 1 58603 007 8, April 2000
[9] Neilson, R. et al , "A discussion of Bandwidth Broker Requirements for Internet2 Qbone Deployment", Work in Progress, 1999
[10] Teitelbaum, B., " QBone Architecture (v1.0)", Internet2 QoS Working Group Draft, http://www.internet2.edu/qos/wg/papers/qbArch/1.0/draft-i2-qbone-arch-1.0.html, August 1999
[11] ITU-T Rec. M.3010, Principles for a Telecommunication Management Network (TMN), Study Group IV, 1996.
[12] ITU-T Rec. X.739, Information Technology - Open Systems Interconnection, Systems Management Functions - Metric Objects and Attributes, 1992.
[13] ITU-T Rec. X.738, Information Technology - Open Systems Interconnection, Systems Management Functions – Summarization Function, 1993.
[14] Multiservices Switching Forum "System Architecture Implementation Agreement" MSF-ARCH-001.00-FINAL, http://www.msf.org
[15] The Parlay Group http://www.parlay.org/
[16] Parlay API's 2.1, Connectivity Manager APIs – Version 1.1, http://www.parlay.org/specs/
[17] The JAIN Initiative http://java.sun.com/products/jain
[18] J. Heinanen, "A Single Rate Three Color Marker", http://www.ietf.org/rfc/rfc2697.txt, RFC 2697, September 1999
[19] J. Heinanen, R. Guerin, "A Two Rate Three Color Marker", http://www.ietf.org/rfc/rfc2698.txt, RFC 2698, September 1999
[20] J. Heinanen et. al., "Assured Forwarding PHB Group", RFC2597, June 1999.
[21] V. Jacobson et. al., "An Expedited Forwarding PHB", RFC2598, June 1999.
[22] X. Xiao and L. M. Ni, "Internet QoS: A big Picture", IEEE Communications, March/April 2000.
[23] S. Blake et. al., "An Architecture for Differentiated Services", RFC2475, December 1998.
[24] "The Next Generation of Network Management" by ©2000 3Com Corporation
[25] "QoS protocols & architectures", White paper, August 1999, http://www.stardust.com
[26] "IP QoS-A Bold New Network", Nortel's White paper, September 1998
[27] http://www.solinet.com/
[28] http://www.violanetworks.com/
[29] http://www.ucl.ac.uk/
[30] http://www.surrey.ac.uk/
[31] http://www.ntua.gr/en_index.htm/, http://www.medialab.ece.ntua.gr/
[32] http://www.ptinovacao.pt/
[33] http://www.grasshopper.de/
[34] http://www.recursionsw.com/products/voyager/