ENABLING DISTRIBUTED QOS MANAGEMENT UTILIZING ACTIVE NETWORK TECHNOLOGY

An algorithm for QoS provisioning using active networks

Stavros Vrontis, Irene Sygkouna, Maria Chantzara and Eystathios Sykas
National Technical University of Athens

Abstract: This paper deals with the QoS management issue utilizing the active network technology. The disadvantages of the centralized network management are presented and the current solutions for distributed network management are discussed. The proposed architecture focuses on the QoS management for DiffServ architecture. The improvements that our architecture offers are presented through a service example, which involves QoS configurations.

Key words: QoS, Active Networks, Distributed Management

1. INTRODUCTION

The widespread growth of the Internet and the development of streaming applications have guided the Internet society to focus on the design and development of architectures and protocols that would guarantee a level of Quality of Service (QoS). QoS is an intuitive concept. The ITU-T Rec. E.800 (1) has defined QoS as "the collective effect of the service performance which determines the degree of satisfaction of a user of the service" or "a measure of how good a service is, as presented to the user, it is expressed in user understandable language and manifests itself in a number of parameters, all of which have either subjective or objective values".

Nowadays, many researchers work on the QoS issue and several architectures and protocols have been designed and implemented. Another research area focuses on the development of APIs that can hide the network elements' details from the service developer. Several relevant platforms have already been implemented, which can provide such APIs, enabling the service developer to configure all network elements. This article analyzes

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35703-4_21
D. Gaïti et al. (eds.), *Network Control and Engineering for QoS, Security and Mobility II*

the problems that developers have to deal with when using these APIs and proposes a solution using the active networks technology. Specifically, in section 1, we present the research activities in the area of QoS management and Services&Network area. Then, we analyze the lacks of the current architectures considering the QoS management in section 2 and we propose a solution through the active networks technology in section 3. Our conclusions and the further work are also discussed in section 4.

1.1 QoS Research

The QoS management (and generally the network management) is rather a centralized issue in the sense that the network administrator configures each node for providing the network users with QoS capabilities. The need for a distributed model, which abolishes the one-to-many logic (administrator-to-nodes), leaded the researchers to improve the network elements by enhancing them with extra functionalities. Currently, the following approaches have been defined for distributed network management through: Mobile agents (MA) (2)(3), Programmable Networks (PN) and Active Networks (AN) (4)(5)(6). These approaches all support computational models that utilize distributed computing resources inside the network. This way, new types of distributed control applications can be developed. The practicability of these paradigms has been demonstrated, mostly based on distributed object systems (e.g. CORBA, Java). Although the original concepts of AN, PN and MA were introduced by different research communities to address different problems, they have started overlapping in focus and applicability, as they are being developed further.

1.2 Services And Network

In today's networks, applications and services are parts of the network operator's domain. With the emergence of mobility and IP, easy creation and rapid deployment of innovative applications that combine different features and critical enterprise data, is a new challenge. Trying to address this challenge Parlay Group and OSA in 3GPP and ETSI are working on the specification of APIs which based on open technology allow applications to access core network functionality (11). These APIs form the interface between what is referred to as the application layer or Service Network on one hand and the core network on the other hand. Applications are positioned in the Service Network and can be deployed independent of the core network and access network that the end-user is using. Although, Parlay and OSA have started from different Fora, they have currently convergence.

JAIN Community is also carrying out related work utilizing Java technology and has form an informal collaboration (12).

2. MOTIVATIONS

In the previous paragraph, we introduced the platforms and APIs for service provisioning, which include the tools for network configuration functions. These platforms provide all the methods to the service developer for producing services that require network configurations or network context gathering. However, there are still practical problems and the current models aren't capable to solve them. In general, even though the provided APIs hide the network complexity from the service developer, he still has to know the network topology (e.g. the IP addresses of the network elements that the service will configure or monitor). Usually, the service provider (SP) utilizes the network that a network provider (NP) offers. Usually, especially for security reasons (Note 1), the NP isn't willing to provide this information to the SPs. Another issue arises if we consider the users' mobility or generally the changes in network topology.

With current platforms, a service developer is able to design a service for a specific network topology. For example, let's consider the following: A videoconference service between two stationary users with QoS. The service developer may design this service using the Parlay APIs. These APIs enable the QoS configurations to all the network nodes through the path between the two users. The service developer needs to know in advance the exact network path between the two users in order to perform the appropriate actions to the relevant intermediate nodes.

Considering the fig. 1, the users A and B have registered to a "QoS videoconference" SP. When they log on, they send their IP addresses to the SP. The SP keeps "usernames-IP addresses" table. User A sends a videoconference request (1) to the SP, asking for a QoS videoconference. The SP takes into account the user's profile and determines the QoS level for the videoconference session. Then, he must configure all the routers, which interfere between the two users. Thus, he must know the exact network topology or he should have mechanisms to discover the path between the two users. Consequently, the NP should give this information or he should allow the SPs to perform "network-scanning" actions. In most cases, NPs are unwilling to provide the network structure information. Moreover, for each SP the network administrator should configure the routers' firewalls to allow network-scanning functions.

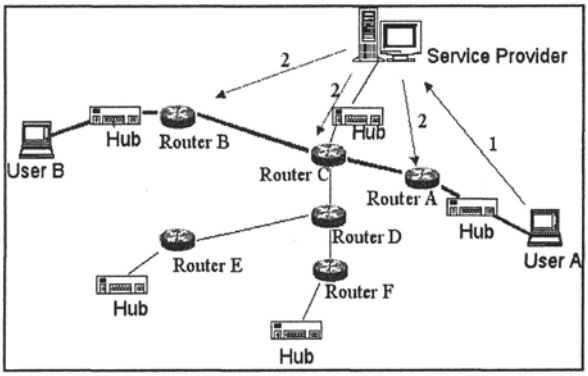

Figure 1. *The videoconference service paradigm*

Considering the case of mobile users (e.g. they access the network via WLAN access points): after the videoconference session establishment, user B moves from the router's B domain and enters the router's E domain. User's terminal's IP address changes (mobile IP is supported) and user B sends a notification (1) to the SP to update its "Usernames-Addresses" table. The SP must know the new path in order to make the new QoS configurations and send the proper messages to each node (2). Hence, he should repeat the initial network-scanning process and the required complex calculations.

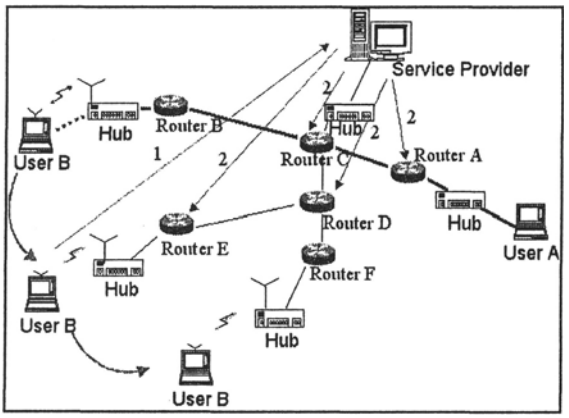

Figure 2. The Videoconference service paradigm with mobile users

3. THE PROPOSED SOLUTION

This article presents an architecture based on AN technology that deals with the mentioned disadvantages of the current platforms. In the next paragraphs, the AN technology basic concepts are introduced, as well as the

first version of the DINA platform that was designed and developed within the CONTEXT IST project (13). The DINA platform is an extension of the ABLE AN platform (14)(15). We will focus on the improvements designed to the ABLE platform in the QoS research area. Moreover, we analyze how the proposed platform could solve the problems that were stated above and present the advantages and improvements that this platform offers.

3.1 Active Networks Concepts And DINA

ANs (16)(17)(18) is a framework where network elements, primarily routers and switches, are programmable. Programs that are injected into the network are executed by the network elements to achieve higher flexibility and to present new capabilities. ANs can work in two different ways: users can preprogram selected network nodes, such as routers, with customized code before running an application. They can also program data packets, which then transport code to nodes along the way to their destination.

DINA is a modular and scalable software architecture that enables to deploy, control, and manage active services, usually called active sessions, over networks entities such as routers, WLAN access points, media gateways and servers that support such services in IP-based networks.

The active node is composed of a Forwarding Element (FE), namely router, WLAN access point, media gateway, etc. with an inherent diverter that detects and diverts active packets to the main separate component, the Active Engine (AE), and the AE, which performs most of the active node's task. The FE and the AE can be either physically separated or collocated at the same machine. The AE consists of the following components: the Session Broker, which receives and parses active packets, handles and manages existing services, and distributes active packets according to requests of the services, the Info Broker which provides an efficient monitoring channel by exporting local state to active sessions and mediates all queries for local state (using SNMP (19)) and the Control Broker, which provides a secure channel for control and configuration operations on the active node. All brokers communicate with the controlled router through the CLI (Command Line Interface) and the OS (Operating System) API. During the progress of the CONTEXT project, we will extend the ABLE platform accommodating new modules (Context Broker, QoS Broker etc.) in order to meet the forthcoming project needs. The modules communicate through UDP transactions and TCP connections.

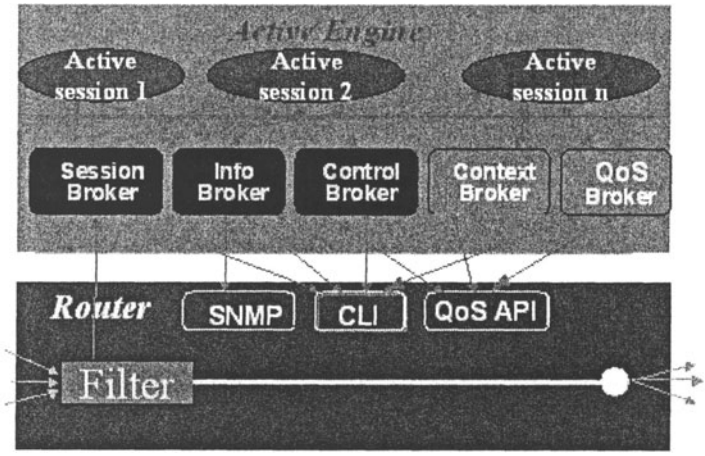

Figure 3. DINA architecture

The communication between two nodes is performed in the following ways: the *topology-blind* addressing mode enables a node to send a packet to the nearest active node in a certain direction, while in the *explicit* addressing mode a packet is sent to a specific active node.

3.2 The QoS Broker

Within the CONTEXT project, we added a separate "QoS Broker" module to the ABLE platform, in order to deal with the problems mentioned in the previous paragraph(s) and improve the QoS functionality that is provided by the current architectures. The active source code triggers the QoS Broker to establish a connection with the controlled router. The ABLE API was extended in order to support QoS actions with the QoS Broker Interface.

The QoS Broker API provides the service developer with a set of methods, which support all the necessary configurations for the QoS setup. The service developers can easily design and implement services without knowing the network topology. The active packets travel through the network and are redirected properly. When an active packet arrives to an active router, it is redirected to the active engine. The encapsulated source code is then executed and the appropriate QoS configurations are performed.

3.2.1 QoS Broker Interface

The QoS Broker Interface provides services with the configuration and the management capability for the routers' network interfaces in order to support QoS functionality. The following QoS configurations refer to the

Differentiated Services architecture (20). The QoS Broker methods appear to Table 1.

Table 1. QoS Broker Interface

Method Name	Description
QoSBrokerInterface	Establishes a tcp connection with the QoSBroker. The commands are passed to the QoSBroker through this connection.
isEdge1	Returns true if the current active node is the edge-1 router.
isEdge2	Returns true if the current active node is the edge-2 router.
isCore	Returns true if the current active node is a core router.
setAFClass	Makes the general configuration for the Assured Forwarding (AF) classes to an egress interface.
setAFxClass	Configures the AFx class to an egress interface. Initially, a general configuration of the AF class is required.
setAFxyClass	Configures the Afxy class to an egress interface.
setBEClass	Configures the Best Effort (BE) class to an egress interface.
setEFClass	Configures the Expedited Forwarding (EF) class to an egress interface.
createAccessList	Creates an access list using the "ipchains" Linux kernel module. An access list is a combination of the source/destination ip address and the source/destination port number for the tcp or udp protocol.
removeAccessList	Deletes a previously created access list.
createClassMarker	Marks the packets of an access list. Marking in this case is considered the setting of the TOS byte of the packets to the appropriate value (DSCP value).
removeClassMarker	Removes a previously created Class marker.
addPolicer	Polices incoming traffic. The policer is attached to an ingress interfaces. Incoming traffic is metered and if it exceeds a specific boundary it is policed.
removePolicer	Deletes a previously created policer to an ingress interface.
monitorQdiscParameters	Returns the qdisc parameters that are set to an interface.
monitorClassParameters	Returns the Class parameters that are set to an interface.
monitorFilterParameters	Returns the Filter parameters that are set to an interface.

3.3 Example Application

The QoS Broker Interface provides to the service developer the tools for performing QoS configurations in DiffServ routers. The SP may utilize the DINA APIs in order to develop efficient applications. We will give an application example for the setup of the QoS settings for the videoconference paradigm between two users. The first user is considered as the source (with IP address: source) and the other user is considered as the destination (with IP address: destination).

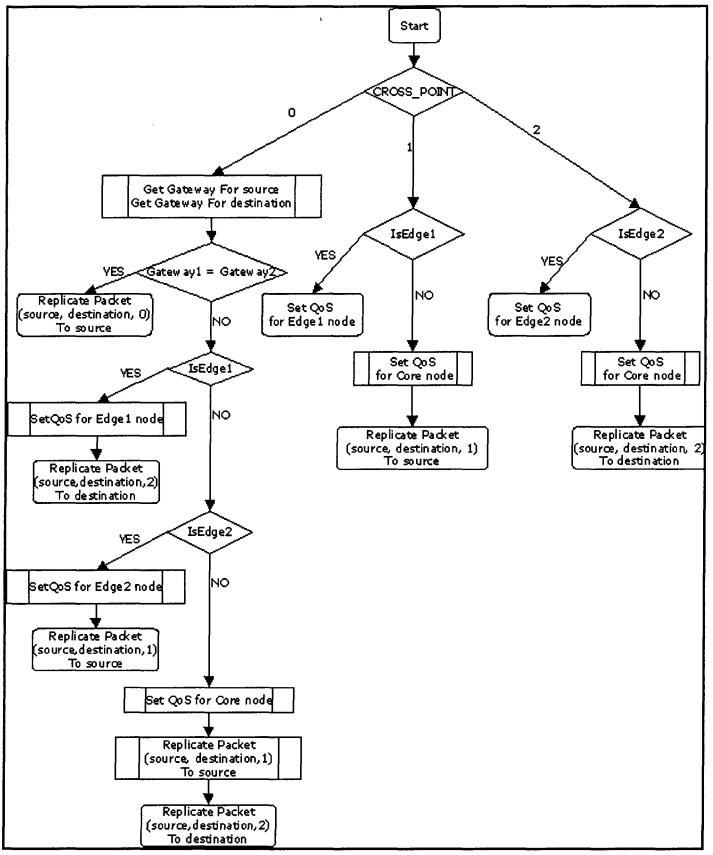

Figure 4. The flow chart of the QoS setup application

The application's source code travels from the server towards the two users. It is captured by the active routers and executed in the active engine's execution environment. The addresses of the users (source, destination) and the CROSS_POINT parameters accompany the application. Each active node receives the application's code and the arguments *(source, destination, CROSS_POINT)*. The application's code during the execution in the active node may cause its replication with possibly different values at arguments, and send it (source code, parameters) towards the proper address.

According to the application's source code, it is checked the value of the CROSS_POINT parameter. If this value is equal to zero, the gateways for the two addresses are attained. If the router has the same gateway for the two destination IP addresses, it means that the packet hasn't still reached a node that belongs to the path between the two users. Hence, the active node simply replicates it with the same parameters. Else, it is concluded that the packet reached a "cross point". It is examined if this node is a core or an

edge node. If it is core, the application replicates itself into two copies and sends them, one for each direction with the value "1" for the packet that travels towards the source and with the value "2" for the packet that travels towards the destination. Also, this node is (QoS) configured as core node. If this node is the gateway for the source, it is configured as the Edge 1 node and the application replicates itself and sends the copy to the destination with CROSS_POINT value "2". Else if this node is the gateway for the destination, it is configured as the Edge 2 node, the application replicates itself and sends the copy to the source with CROSS_POINT value "1". Finally, if a packet with CROSS_POINT value "1" reaches an active node, it means that this node belongs to the path between the users, has already passed the cross-point node and travels towards the source. It is checked if it is core or the edge 1 node and the router is configured accordingly. In core router's case, the packet is replicated and sent with the same parameters towards the source. Similarly, if a packet with CROSS_POINT value "2" reaches an active node, it means that this node belongs to the path between the two users, has already passed the cross-point node and travels towards the destination. It is checked if it is a core or the edge 1 node and the router is configured accordingly. In case of core router, the packet is replicated and sent with the same parameters towards the destination.

3.4 Applying The QoS Broker APIs

Next, we examine the proposed solution reflecting the videoconference service example on top of AN(Fig.5). All routers are controlled by DINA active engines extended with the QoS Broker module.

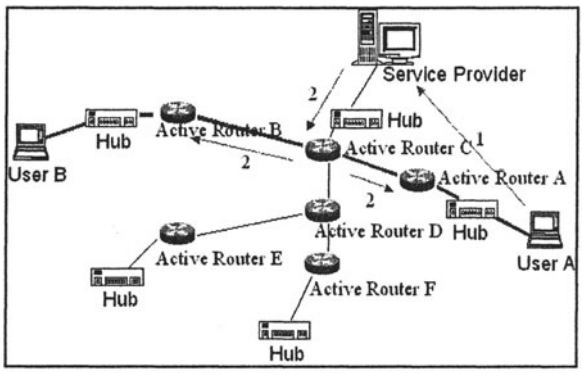

Figure 5. The videoconference service paradigm with ANs technology

When the SP receives a request for the setup of a videoconference between user A and B (1), it generates an active packet, which contains the

source code of the application example of the previous paragraph. The first active node (router C) captures the active packet. The source code in the active packet causes the QoS configuration of this node and the generation of two active packets (2), which are directed to active router A and B respectively. These packets perform the QoS setup to these routers.

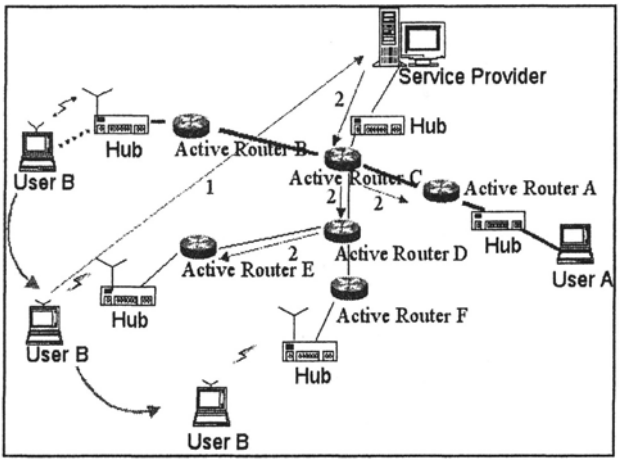

Figure 6. The videoconference service paradigm for mobile users with ANs technology

Now, let's examine the second case with the videoconference service between two mobile users with QoS. User B moves to the active router's E domain and sends a notification (1) to the SP in order to inform him that his IP address has changed. The SP sends the same active packet (2) with the new address in the user B address field. The smart source code is executed in the first node (router C) and the proper QoS actions are performed. Also, two active packets are generated and sent to nodes A and D (2). The source code in router D also generates a new packet. This packet is sent to router E (2).

Using the ANs technology and specifically the QoS Broker module, we achieved the following improvements:

1. *Fast Deployment:* When a user sends a request for QoS to the SP, the SP simply generates an active packet. Without using the ANs technology, the SP would have to discover the path between the two users by sending messages to the involved users and then by performing calculations using complicated algorithms. The SP would also have to send the QoS configuration commands to each router. This procedure delays considerably the service deployment.
2. *Network invisibility:* Using the ANs technology, the SP of our example can perform network operations for a network path without knowing the included nodes. This way, the NP can keep the network

infrastructure hidden from the SPs. Furthermore; the SPs aren't required to keep stored network structure information and to be overloaded with data storage/update activities.

3. *Efficient messages exchanges economy:* With ANs, the messages that are exchanged in the data link layer are significantly reduced. Considering our service example with the static users, the service should configure 3 nodes. Using the AN platform with the application, which discussed in previous paragraph, totally 3 packets should be sent, while using the traditional pathetic network, this number arises to 5 messages. In case of the service example with the mobile users, the service must configure 4 nodes. Within the AN platform 4 messages are required, while within the non-active network: 8 messages. Generally, in case of the setup of n nodes, n messages are required using the AN, while the number of the required messages increases significantly using non-active network and depends on the network topology. Further discussion in this issue is out of the scope of this article.

Table 2. Messages economy with the AN platform

Nodes	Total messages for Layer 2 (non-active)
3	5
4	8
n	$\sum_{n} hops_i$ ($hops_i$: number of hops between SP and i-node)

4. *Distribution of consumed resources:*All the calculations are performed in the active engines and not only to the SP. This way, we can achieve the distribution of the required calculations to all the relevant nodes and the SP's decongestion.

4. CONCLUSIONS AND FURTHER WORK

In this article, we presented the basic QoS concepts, the efforts of the researchers to develop APIs that hide the network complexity from the service developer. The current architectures still need improvements. The challenges to face and the proposed features that the network should include were discussed. On this basis, we extended the ABLE AN platform by adding a QoS Broker module. Furthermore, we analyzed the advantages of this platform and explained how this technology can face the shortcomings of current architectures and improve the network services in the QoS area.

Currently, we are considering the possibility to perform measurements and provide performance evaluation tests using simulation in order to compare our architecture to the traditional non-active platform concerning

the issues of the fast deployment, the number of messages' exchanges and the amount of consumed resources in the network nodes. Theoretically, it is expected that our platform's performance in relation with the current platforms will increase analogically to the number of the network nodes.

We also intend to deal with the design and implementation of smart applications that utilize the QoS Broker API and perform more complicated QoS configurations (for example considering the videoconference service for N users). Furthermore, the QoS Broker module refers to the DiffServ architecture. We could also consider the design and implementation of an IntServ QoS Broker (21)(22). Finally, we intend to continue our research on more complicated networks that comprise of several interconnected DiffServ domains (with Service Level Agreements for each pair of domains) or even of heterogeneous networks with IntServ and DiffServ domains.

NOTES

1. The proposed solution utilizing ANs seems paradoxical in the sense that our solution considers that the NP allows the SP to execute source code to the active nodes, situation that seems quite insecure. However, the DINA execution environment contains a security manager component that controls the application's functionality. Specifically, the application is permitted to perform actions only using the DINA's APIs. For example, in case of this article's application, the user has to utilize the QoSBrokerInterface API in order to perform QoS configurations to the network elements. The application may contain several functions that cause the QoS configuration (e.g. tc commands) or other activities (e.g. file access), however the security manager will not allow their execution. Moreover, in our case, the QoSBrokerInterface methods pass the commands that will be executed to the QoS Broker module. The QoS Broker module parses the commands and decides to permit or not their execution (e.g. if the allocated bandwidth for the EF class is greater than the total available network interface's bandwidth deny the execution of this command). Of course, several improvements could be considered (e.g. a more advanced QoS Broker could communicate with a database with SPs' profiles in order to perform authentication, authorization and accounting activities).

REFERENCES

1. ITU-T Rec. E.800, "Terms and Definitions Related to the Quality of Telecommunication Services", *Blue Book*, 1988.
2. La Corte, A. Puliafito, O. Tomarchio "QoS management in programmable networks through mobile agents". Microprocessors and Microsystems, 25(2): 111020, April 2001.
3. H. de Meer, A. Puliafito, J.P. Richter, O. Tomarchio. "QoS-Adaptation by Software Agents in the presence of defective Reservation Mechanisms in the Internet". In 3rd IEEE Symposium on Computers and Communications (ISCC'98), Athens (Greece), June 1998.

4. D. Raz and Y. Shavitt. "An Active Network Approach for Efficient Network Management". *IWAN'99*, July 1999, Berlin, Germany.
5. D. Raz and Y. Shavitt. "Active Networks for Efficient Distributed Network Management". *IEEE Communications Magazine*, 38(3):138--143, March 2000.
6. R.Kawamura, R.Stadler: "A middleware architecture for active distributed management of IP networks", *in Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS 2000)*, Honolulu, Hawaii, 10-14 April 2000, pp. 291-304
7. Di Fatta G., Gaglio S., Lo Re G., Ortolani M., "Adaptive Routing in Active Networks", IEEE Openarch 2000, Tel Aviv Israel 23-24 March 2000.
8. M. Tasir A.R. ,Linge N. "Intelligent Active Routing for Supporting QoS Demands", 3rd Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, PGNet 2002. EPSRC, Liverpool John Moores University, ISBN 1 902560 086, pp52-56.
9. T. Faber, ACC: 'Active Congestion Control", IEEE Network, May/June 1998, pp.61-65
10. N.Prabhavalkar,M.Parashar, P.Agrawal, LGC: "An active Congestion Control Mechanism"
11. A. Moerdijk and L. Klostermann, "Opening the Networks with Parlay/OSA: Standards and Aspects behind the APIs", to appear in IEEE Network Magazine, www.parlay.org
12. J. de Keijzer, D. Tait, and R. Goedman, JAIN: "A new approach to Services in Communication Networks", IEEE Communications Magazine, January 2000.
13. The Context IST Project: "Active Creation, Delivery And Management Of Efficient Context Aware Services", IST-2001-38142-CONTEXT, http://context.upc.es
14. ABLE: the Active Bell Labs Engine, http://www.cs.bell-labs.com/who/ABLE/
15. ABLE - The Active Bell-Labs Engine for Network Management" demonstration at OpenArch'99, New-York, NY, March 1999.
16. "Towards an active network architecture", D. L. Tennenhouse and D.J. Wetherall, *Computer Communication Review*, 26(2), April 1996.
17. David.L.T, Smith.J.M, Sincoskie.W.D, David.J.W. "A Survey of Active Network Research, IEEE Communications magazine", Vol.35 No.1, pp80-86, January 1997
18. "Directions in active networks", K.L. Calvert, S. Bhattacharjee, E. W. Zegura, and J. Sterbenz, *IEEE Communications Magazine*, 36(10):72–78+, October 1998.
19. SNMP research, http://www.snmp.org/
20. K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC2474, Dec. 1998.
21. R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, June 1994.
22. Active Reservation Protocol Home http://www.isi.edu/active-signal/ARP