An Active Network Approach to Support Multimedia Relays

Manuel Urueña[1], David Larrabeiti[1], María Calderón[1], Arturo Azcorra[1], Jens E. Kristensen[2], Lars Kroll Kristensen[2], Ernesto Exposito[3], David Garduno[4], and Michel Diaz[4]

[1] Universidad Carlos III de Madrid. 20, Av. Unidersidad, Leganés 28913, Spain
{muruenya,dlarra,maria,azcorra}@it.uc3m.es
[2] Ericsson Telebit A/S. Skanderborgvej 232, DK-8260 Viby J., Denmark
{jens.kristensen,lars.k.kristensen}@lmd.ericsson.se
[3] ENSICA, DMI 1 Place Emile Blouin 31056, Toulouse Cedex, France
ernesto.exposito@ensica.fr
[4] LAAS CNRS. 7, Avenue du Colonel Roche, 31077 Toulouse, France
{dgarduno,diaz}@laas.fr

Abstract. This paper summarizes a pragmatic technical approach to active networks that supports prototyping of multimedia transport protocols. One of the targets of the architecture defined is the extension of the processing capabilities of existing routers trying to keep up the forwarding performance of regular packets. This work presents practical results obtained from the development of a Java execution environment based on the design principles presented and proposes a method for the transparent deployment of multimedia relays over a source-destination path, tested in several experiments.

1 Introduction

Processing multimedia flows inside the network is known to be a rather convenient way to support multi-QoS (reflectors) or multi-coding (translators) in multipoint sessions. Usually the approach followed is to locate such relays in servers. This has the major drawback that the users usually have not the ability to choose the optimum place to run these services (usually this optimum place is in the heart of the network) nor the possibility to adapt the server behaviour to their specific needs (new codecs, specific bitrates, partial reliability, etc). The former problem can cause suboptimal delay and bandwidth usage, and the latter, simply prevents the development of new ways of processing multimedia flows, if the server is not open to this possibility. This could be overcome by bringing in to scene some concepts developed in the light of the active network paradigm, namely, actually locating relays on the nodes on the source-destination path rather than on a remote off-the-way host. The main drawback for this is that multimedia flow processing is a CPU-intensive task and currently available routers are not designed for such work.

This paper summarizes the experience obtained from a design process aimed at building an active network framework useful for new multimedia services. This framework was developed and applied to multimedia processing in the context of the IST project GCAP [1] [5] [6]. Two outputs of this project were directly related to active network support of multimedia flows. One of them was a software/hardware prototype designed to demonstrate the way commercial routers could be easily enhanced to support a few active processing capabilities. The second one was the practical application of this system to support the deployment of multimedia multicast protocol entities running as active applications.

This paper has the following structure: the second section reviews the currently available Java extensions for multimedia applications over IP and its relationship with active networks. The next section describes SARA, an architecture defined to integrate only practical principles from active networks technology, including innovative solutions to keep most router's architecture unchanged. This third section also includes a description of a prototype developed to demonstrate the feasibility of this approach. The fourth section discusses a case study to apply this framework to the deployment of multimedia processing, in particular to multimedia relays. Finally, some conclusions are drawn.

2 Java for Networked Multimedia Applications

The Java language [2] developed by Sun Microsystems has achieved a noticeable success among application developers. Initially marketed as "the Internet language" with its "Write Once, Run Anywhere" slogan, it was employed to develop "applets", small programs embedded in web pages to be downloaded and executed by browsers. This phase lost momentum and most of current Java applications do not reside in desktops but in servers, as Java, plus XML technologies as SOAP, turned to be the ideal platform to develop web applications and services.

Standard Java libraries have been in continuous growth and nowadays includes many packages that ease multimedia development:

- Java Media Framework (JMF) [3] is an extension package to develop multimedia applications and services using the Java language. This package is cross-platform as it has been implemented in pure Java. However there are some other implementations for widely deployed operating systems (Windows, Solaris and Linux) with native code to enhance capture and playback performance.

[5] This work has been funded by the IST project GCAP (Global Communication Architecture and Protocols for new QoS services over IPv6 networks) IST-1999-10 504. Further development is being supported by the Spanish MCYT under project AURAS TIC2001-1650-C02-01.

[6] Disclaimer: this paper reflects the view of the authors and not necessarily the view of the referred projects.

– The Real-Time Specification for Java had been also released [4]. It defines the `javax.realtime` package that extends the Java language so it can be appropiate to develop real-time applications by adding predictable scheduling models, memory management, synchronization and resource sharing, asynchronous event handling and many others mechanisms needed by real-time applications, as multimedia processing ones.

However, Java has been designed as a high level language, that is, to develop applications, not to offer a system programming platform, although some efforts as the above mentioned Real-Time Specification may change that in the next future. Moreover, Java programs must be able to run in a variety of platforms from different vendors, thus only the common features of all platforms can be offered by the standard API.

These factors had led to a limited Java networking API. The standard library version 1.4, recently released, supports IPv4, IPv6, TCP, UDP and multicast sockets. Enough for most of networked applications but quite constrained for low-level advanced network programming. For example, the current API provides socket options but IP options still cannot be set. During the community process that led towards the latest release, raw socket support was discussed but it was later dropped off from the final specification due to security considerations.

To overcome the lack of several features needed for the development of advanced multimedia services and, in particular, the target packet processing capabilities of the active network platform here described, a new Java networking library was needed. That is the motivation of a raw socket library, which is described later in this article.

Due to its multiplatform design, Java has always been heavily associated with mobile code, such as agents and active applications. That feature, absolutely vital in such an heterogeneous environment as an active network, is able to overtake other Java drawbacks such as its limited networking API and lower performance. This, together with Java's enhanced multimedia capabilities, made it the choice to prototype the system described in the next section.

3 Simple Active Router Assistant Architecture

As already introduced, our aim is the definition of a pragmatic framework to implement basic active networking functions, valid both for IPv4 and IPv6, and realistic in an industrial context. This section defines this framework which we will refer to in the remainder of the paper as the "router-assistant" approach. This way of building active networks is based on a set of techniques selected according to its industrial applicability.

3.1 Motivations

The key features of the defined framework and the rationale behind them are:

- **Current Routers have been designed to forward packets, no to process them up.** Most high speed commercial routers have a distributed architecture controled and supported by a CPU with a capacity far below todays hosts' capacity; in principle, they needn't have it because a CPUs main function is system management and routing, and most of a router's performance is due to its interface processors specialised in packet forwarding. Therefore, a router CPU cannot spare so much horsepower itself to CPU-intensive processes as e.g. multimedia transcoding is. Thus an Execution Environment with Active Applications cannot be run without disrupting forwarding performance for both, active packets and 'regular' ones.
- **Can you add capacity without buying a new router?** Active Applications should, by definition itself, be able to make a broad set of operations, from traffic probing to multimedia flow transcoding. Therefore resource requirements of active nodes should be able to be changed in a very dynamic fashion. Scalability must be considered as a primary issue in order to evaluate active network proposals.
- **Internet is not active.** Most of current Active Networks platforms rely on overlay networks, that is, building an parallel active Internet [5] with its own routing protocols. Clearly this effort requires a lot of work to be added to the development of active network entities, a quite complex task by itself. The experience with current middleware devices (NAT, firewall,...) shows that most intelligent services should be transparent to the end user. Therefore, explicit addressing, as overlayed active networking is, should be allowed but not mandatory.
- **Active Capsules poses a great security risk.** Capsules, that is, active packets carrying the code to process them, allow end users to inject its own protocols. This mechanism provides the greatest level of flexibility in protocol deployment ever seen for the price of security checking. Mobile code must be checked for security and safety reasons before being executed. Many mechanisms have been proposed, from signed code [6] to safety checking [7], each one has advantages and drawbacks but none has been able to fulfill all requirements. Capsules cannot be justified as they are inherently harmful, as they could compromise the security of the whole network. This problem limits in practice the degree of dynamism of active code.
- **Active Applications must be dynamically deployed... but what is dynamic enough?** Current network services and protocols are 'static'; the deployment of a new protocol is really a painful task (IPv6 is a perfect example). Preventing this problem is one major objective of the active network community and the reason for the theoretical capsules approach. However, dynamic code deployment requirements can be lowered considering that service and protocol deployment are not tasks to be done on a daily basis.

3.2 Design principles

Considering these facts and introducing practical constraints, an architecture named Simple Active Router Assistant Architecture (SARA) has been defined

with the purpose of giving a pragmatic approach to the most important services provided by active networks.

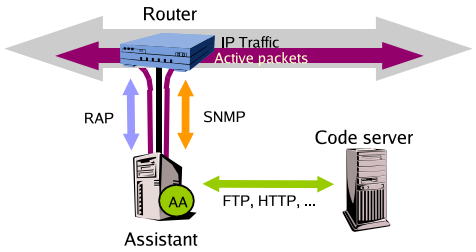

Fig. 1. Simple Active Router Assistant Architecture.

A summary of architecture characteristics follows:

- **Active node = Router + Assistant.** Legacy routers can be upgraded to active nodes by delegating active processing to an attached host (or host cluster) called Assistant via a high speed interface. The assistant host runs the NodeOS, Execution Environment and Active Applications, thus avoiding such process to consume router's resources. This architecture allows administrators to increase active processing capacity by adding more assistants, thus adapting nodes to the resource requirements of active applications deployed on them.
- **Router diverts packets to Assistant.** In order to adapt easily routers to this architecture, its requirements are reduced to simply forward certain packets to its assistant, in particular those defined as active. Hence, router's performance cost is limited to just to identifying such packets. To accomplish this task without a severe performance penalty, the Router Alert IP option [8] [9] should be employed. This option forces routers to get the packet off the fast-path and deliver it to the appropriate software entity to handle it.
- **Transparent active processing.** As it can be inferred from the previous paragraphs, packets to be processed are not explicitly addressed to active nodes but captured as they are forwarded to their final destination. An overlay network of active nodes is not necessary as they are truly integrated with real network nodes. Adding an active node to the network requires no routing protocols other than the one running on the network itself. Of course, active nodes may be made aware of other active nodes in the network and build an overlay if necessary, to override the routing tables. Nevertheless transparency is a desirable property for the sake of end-system simplicity.
- **Assistant processes diverted packets.** Packets diverted by the router are dispatched by the assistant's Execution Environment and put on the input queue of the corresponding active application. An active application can alter anything in the packet from network to application level and may reinject the packet to the router so that it can be routed as any other packet.

- **Dynamic Active Application deployment.** Any code loading/execution approach can be applied under SARA architecture. However, the recommended method is that active packets carry references to active code that processes them (and security credentials if necessary). If code has not been yet loaded it can be retrieved on demand from code servers by existing methods (e.g. ftp), and then loaded dynamically onto the Execution Environment as a new active application.
- **Monitoring the router state.** One of the drawback of this approach is the need for greater interaction complexity between router and assistant as they reside in separate hosts. This communication is necessary to provide active applications with the sense of actually being run on the router. In order to make it possible to adapt the active behavior to the real-time network status, active applications need to access the router's state. As one of SARA objectives is to cooperate with legacy routers, the Simple Network Management Protocol (SNMP) [10] was selected, as it is the standard management protocol, supported by most of network devices. However, a more efficient management protocol could be used instead, to avoid some problems associated to SNMP. In order to avoid the router performance hit when the router's state is constantly polled. A small cache, managed by the Execution Environment, storing recent state variables may improve both router performance and applications response time.
- **Router-Assistant Protocol.** Two conformance levels have been defined for routers supporting SARA. Level 1, for routers that only divert active packets and accept SNMP queries, and level 2, for routers that also support Router-Assistant Protocol (RAP) operations. This protocol allows assistants to configure somehow its attached router. Some examples of RAP operations may include setting tunnel endpoints or configuring firewall rules. One of the most important operations requested by assistants to level 2 conformant routers is to divert non-active flows as it is done with active ones. This mechanism enables active applications to capture and process non-active packets; that is, to enhance communications without modifying current applications and services. At this scenario active packets are relegated to the control plane.
- **Trusted code.** From the authors' viewpoint, end users freely injecting mobile code in the network is not a realistic scenario. Instead of evaluating mobile code safety techniques, a more pragmatic approach has been taken. Each network administrator must be able to control exactly what active applications are allowed to execute into his domain and never load untrusted ones. In this scenario trusted active applications may be provided by the network administrator itself, by customers or by any trustable third party. Active Applications stored in code repositories of a certain administrative domain are allowed to execute in that domain as they have trusted code. Code safety can be checked in advance by multiple techniques as auditing source code or just trusting the application supplier. In fact, this is the common procedure of today's mobile code, as it is the casew of software packages

downloaded from internet: applications from known vendors are trusted and installed directly.

- **Latency.** SARA architecture is not well suited to support applications that require a tight interaction with low level router resources (e.g. selective dropping of packets buffered at the router queues). Although latency might be reduced by increasing the bandwidth of the link between the router and the assistant, the fact is, that most diverted packets need to be routed twice. Depending on the platform chosen, this may be an important issue for some real-time multimedia applications. However, it should be noted that a similar delay is found when we choose the alternative server-based approach, as packets are forwarded to the server and sent back to the network once processed.
- **Active processing isolation.** As the Execution Environment and the Active Applications are executed on a separate host, the router is quite shielded from many kinds of active processing errors. In the worst case scenario a buggy active application could crash the assistant host, and diverted packets may get lost; however any other flow passing through the router will remain unaffected. To reduce the packet loss due to assistant's crash, the RAP protocol defines a heartbeat mechanism to detect when an assistant has become unreachable.

3.3 Implementation

To demonstrate the feasibility of the proposed architecture, a SARA prototype has been developed [11]. Two testing platforms are available today. One fully based on linux (playing both roles: router and assistant as a development/testing scenario) and a hybrid platform where the router used is an Ericsson-Telebit AXI462 running a modified kernel adapted to interwork with an active assistant. As already mentioned, a main goal of this latter platform, currently under conformance level 1, is to demonstrate that it is possible to build an active network platform based on commercial routers without a significant drop of performance on regular packets.

Active applications and the developed execution environment are based on Java. However, given the limitiations of the current JVM interface reviewed in section 2, a small native extension was needed. In particular, `java.net` package does not contain any object that allows Java applications to open raw sockets or set IP options. Both characteristics are needed by SARA prototype, the first one to capture active-packets while the second one is needed to add the router alert option to active packets. To fulfill these requirements, a networking library for Java providing support for IPv4, IPv6, UDP and TCP sockets with advanced options was developed. This library defines a number of classes, resembling standard API ones, that actually wraps system calls to the standard BSD C socket API, using the Java Native Interface (JNI). Multiplatform capability is somewhat kept as SARA native code is small and easily portable.

Although the current implementation is still at an early development stage and there is much room for performance improvements, the basic throughput

measures look promising. As it can be seen in figure 2, flows up to 10 Mbps of mid-sized active packets can be processed by the current SARA prototype.

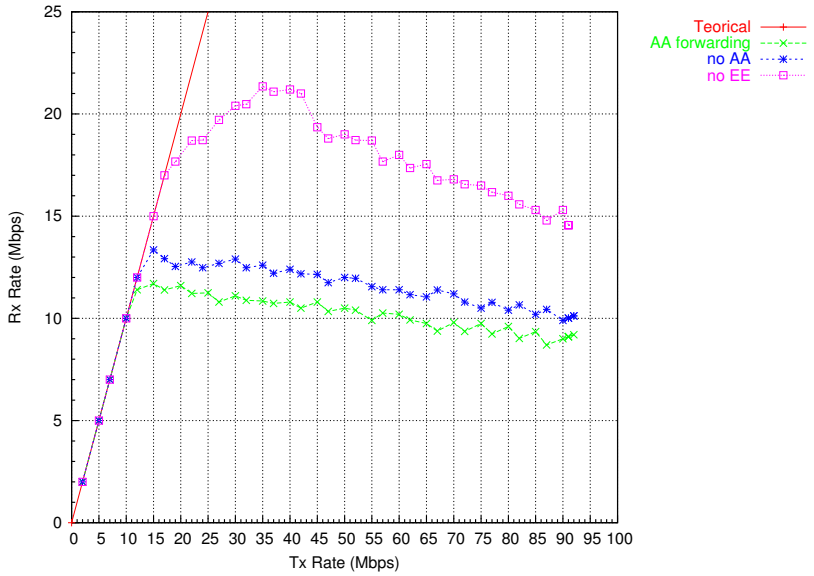

Fig. 2. Effective throughput for EE and simple AA with 1024-byte packets.

4 Multimedia relay deployment

Several authors have highlighted the benefits obtained from applying the active network paradigm to multimedia flows. Many research initiatives propose the deployment of active relays inside the network to provide solutions as transcoding [12] [13] or multimedia multicast [14] [15]. However the problem related to where these proxies must be deployed has been scarcely addressed.

4.1 Relay location

First of all, the "right places" for relay deployment need to be studied. As a rule of thumb, it can be said that network nodes with some sort of discontinuity are ideal points to set up a multimedia relay or, in general, an active entity. Its behavior depends upon what kind of gap is found. A list with some examples:

– Border routers or ISP gateways. This is the most obvious place, and many existing technologies as NAT or firewalls need to be deployed into such nodes. An active node deployed at an edge device could build automatic tunnels to forward multicast flows whenever its ISP does not provide multicast service.

– Bandwidth discontinuity, as a router with different speed interfaces, such as a pool of modems, an ATM interface with several channels configured with different service categories and rates, etc. This can be a good candidate to deploy for example a multimedia codec translator.
– Lossy links, as wireless access segments, and high delay links, as satellite segments. Buffering packets until the corresponding acknowledgement is received prevents longer delayed end-to-end retransmissions, in fully and partially reliable transport services.
– Nodes involved in a muticast delivery tree branching to several destinations, specially in the case of multi-QoS support. Network processing in these nodes could resolve many problems of reliable multicast protocols as ACK/NACK implosion.
– Congested nodes. Selective packet discarding could dynamically adapt multimedia flows traversing congested links to achieve a better quality, for example by dropping video frames and priorizating sound packets.

Many of these discontinuities are rather static, therefore the requirement to deploy a proxy dynamically is lowered as active applications can be continuously running at such places. However, some others services, as automatic tunnels, multicast nodes or congestion spots have a very variable nature, and hence relay deployment must be more flexible, and the framework that supports such proxies should be able to detect the best place to run these applications. Active networks can provide such deployment framework, as well as the mechanisms to compute optimal relay placement.

4.2 Relay deployment methods

By definition, all existing active network platforms provide dynamic code deployment, however not every platform may be suitable to detect the optimum active location, mainly due to the need of defining an overlay network. Overlay networks hide underneath path characteristics as all links between two active nodes are aggregated as a virtual circuit. Another problem is the deployment of short-lived proxies, as the overlay network needs to be reconfigured to add new nodes.

SARA architecture allows the transparent deployment of active entities while avoiding these problems. As SARA does not require defining an overlay network, overlay routing is not an issue, and, thanks to SNMP, an assistant could have an influence area around its attached router. By setting SNMP traps, active applications can react more quickly to network changes. SARA ability to inject any kind of IP packet could be used to monitor flow paths by sending pings to check response times, packet loss or any other parameter interesting for active applications. For instance, tools like *A-clink* [16] can be very useful to optimize the placement of multimedia proxies. This active application is a path characterization application based on the freely available *clink* tool, enhanced with active support to improve the efficiency and accuracy of estimations yielded by existing end-to-end performance estimation tools such as *pathchar*, *pchar*, *clink* and *nettimer*.

5 A case study

While the previous section refers to theoretical aspects of proxy deployment using active networks, the following describes an experiment carried out in the context of GCAP, with a real active application running over a SARA implementation. Firstly, we describe the active application involved: a proxy running the FPTP protocol, and then the setup is described.

5.1 Fully Programmable Transport Protocol (FPTP)

FPTP is transport protocol sub-layer located between the application and the traditional transport protocols. Multimedia applications are able to access to the FPTP API directly (FPTP sockets) or using an XML-based, QoS configuration. In both cases applications have to specify the QoS transport required in terms of order, reliability, bandwidth, time and synchronization constraints. FPTP protocol is able to instantiate standard transport protocols or to deploy new protocol mechanisms and services, following the QoS transport constraints specified by the application.

FPTP protocol consists of 2 kinds of transport connections:

- The multimedia connection control allows communicating both sides of the multimedia connection. This connection is a bidirectional connection and based on XML messages exchanged (SOAP).
- The monomedia connections allow transferring the media data following the QoS constraints required. Each monomedia connection is a unidirectional connection.

A FPTP monomedia session is established between a sender and a receiver entity. In both sides the transport protocol mechanisms can be instantiated and/or programmed according the QoS constraints required. If the QoS constraints can be assured by the standard transport protocols, the FPTP monomedia connection will be instantiated using these available protocols. If the requirements are applications specific, the transport protocol services can be instantiated using the available FPTP transport protocol services or programming and deploying new mechanisms and services.

5.2 Deploying FPTP proxies with SARA.

An experiment was held to stream video between two sites connected across the Internet. The FPTP protocol was employed to adapt video quality to QoS offered by such a connection. As the video server and the client were executing traditional RTP multimedia applications, intermediate proxies were needed to issue the QoS enhancements. The SARA prototype did provide active network support for such proxy deployment as well as path QoS characterization between sites.

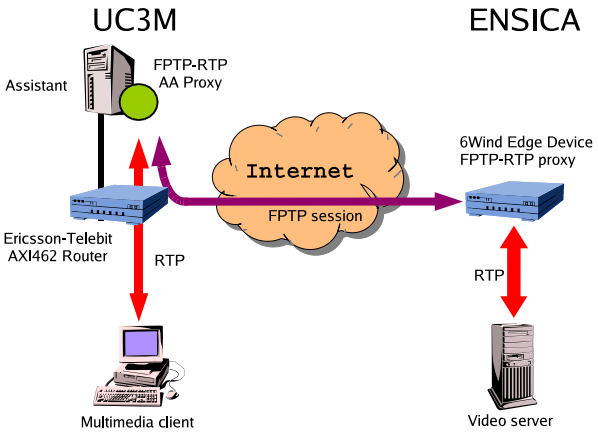

Fig. 3. FPTP-RTP proxy deployment.

The video server run at Ensica (France) while the client application was running in Universidad Carlos III de Madrid (Spain). Proxies were deployed over two different platforms, a 6Wind edge device located at France and an active node formed by a Ericsson-Telebit AXI462 router and a linux box running SARA Execution Environment where the FPTP-RTP proxy was automatically loaded on demand by downloading code from a local server via HTTP.

As seen in figure 3, three sessions were established, an traditional RTP session between server and the French proxy, a FPTP session between both proxies and another RTP session as the Spanish proxy received FPTP packets and sent the video flow to the client using RTP. An asymmetrical deployment method was used (transparent and explicit deployment respectively) and the experiment succeed in demonstrating cross-platform interoperation of FPTP proxies and dynamic code deployment thanks to SARA platform.

6 Conclusions

This paper has reviewed the preeminent position of Java as a multiplatform language to develop active network applications, along with the multimedia framework it provides. Both features converts Java platform in the ideal tool to develop active applications to handle multimedia flows, in spite of some drawbacks as performance and lack of low level communication facilities. That is why Java was the chosen language to implement the Simple Active Router Assistant Architecture (SARA).

SARA is an architecture that allows upgrading legacy high-speed routers to work as active nodes by delegating active processing to an external entity called assistant. This design is based on a set of existing network technologies, that were selected because they could be applied in a production environment oriented to multimedia flow processing. The experience has shown that, such a pragmatic

approach is a suitable way to support multimedia relays, and also to compute accurately the optimum places to deploy them, a problem partially addressed nowadays. Transparent processing of active packets is the most important feature claimed by this system. It avoids the need of defining an overlay network as many other active network platforms have to do. To accomplish that task, a new library has been designed to overtake certain limitations of Java networking API.

The feasibility of SARA architecture has been demonstrated by running several multimedia experiments employing active FPTP-RTP proxies. These entities ran over a prototype of SARA assistant working together with a commercial router. The router was an Ericsson-Telebit AXI462, slightly modified to divert active packets containing the router alert IP option, both for IPv4 and IPv6.

References

1. GCAP IST project home page: http://www.laas.fr/GCAP
2. J. Gosling, B. Joy, G. Steele: The Java Language Specification. Addison-Wesley 1996.
3. Java Media Framework home page: http://java.sun.com/products/java-media/jmf/index.html
4. G. Bollella et al.: The Real-Time Specification for Java. Addison-Wesley 2000.
5. Abone home page: http://www.isi.edu/abone/
6. L. Gong, R. Schemers: Signing, sealing, and guarding Java objects. In Mobile Agents and Security, volume 1419 of Lecture Notes in Computer Science. Springer-Verlag, 1998.
7. P. Fong, R. Cameron: Proof linking: An architecture for modular verification of dynamically-linked mobile code. Proc. of the 6th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE-6), pages 222-230, Orlando, Florida. November 1998.
8. D. Katz: IETF RFC 2113: IP Router Alert Option. February 1997.
9. C. Partridge, A. Jackson: IETF RFC 2711: IPv6 Router Alert Option. October 1999.
10. J. Case, M. Fedor, M. Schoffstall, J. Davin: IETF RFC 1157: A Simple Network Management Protocol (SNMP). May 1990.
11. SARA home page: http://enjambre.it.uc3m.es/ sara.
12. E. Amir, S. McCanne, R. Katz: An Active Service Framework and its Application to Real-time Multimedia Transcoding. Proc. of ACM SIGCOMM '98. 1998.
13. I. Kouvelas, V. Hardman, J. Crowcroft: Network Adaptive Continuous-Media Application Through Self Organised Transcoding. 1998.
14. B. Banchs, W. Effelsberg, C. Tschudin, V. Turau: Multicasting Multimedia Streams with Active Networks. Proc. of the 23rd. Annual Conference on Local Computer Networks. 1998.
15. S. Kang, H. Yong Youn, Y. Lee, D. Lee, M. Kim: The Active Traffic Control Mechanism for Layered Multimedia Multicast in Active Network. Proc. of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. 2000.
16. M. Sedano, B. Alarcos, M. Calderón, D. Larrabeiti: Caracterización de los enlaces de Internet utilizando tecnología de Redes Activas. III Jornadas de Ingenería Telemática. Barcelona. September 2001.