

MULTIMEDIA MULTICAST PROTOCOLS BASED ON MULTIMEDIA MODELS

MICHEL DIAZ, DAVID GARDUNO, THIERRY GAYRAUD, STEPHANE OWEZARSKI

*LAAS CNRS, 7, Avenue du Colonel Roche, 31077 Toulouse, France
Email: diaz@laas.fr, dgarduno@laas.fr, gayraud@laas.fr, sowe@laas.fr*

Abstract. - In multimedia distributed applications such as videoconferencing, real time data exchange requires formal models to design and specify the Quality of Service (QoS) of such multimedia communication. The proposed multimedia multicast protocols are based on enhanced Timed Petri Nets multimedia models.

These models allow synchronised data streams to be exchanged in partial order and reliability, according to the required QoS. So, using a multicast monomedia transport sub-layer, we are able to design and implement multicast multimedia transport protocols, in which the multimedia synchronisation is defined in our multimedia models. It will be shown how these protocols have been designed to support multicast communication.

All these protocols must be deployed in current networks, without operator support. This deployment will be done using active networking facilities to download the required code remotely on selected nodes.

Experiments have given the first implementation results on European network as defined in EU project "GCAP".

Keywords – Multimedia model, multicast, QoS, Active networking, videoconferencing.

1 Introduction

In future multimedia multicast applications, time-critical data such as audio and video information must be exchanged between communicating entities.

For this purpose, advanced high-level communication protocols are necessary needing a major design and development effort. With today's approaches, such an effort is part of the application development. EU-funded GCAP project proposes a global approach that will make advanced designers, users and applications able to define and use new protocols to develop and rapidly deploy new optimised and innovative architectures.

To address the needs of multimedia applications, GCAP is targeting a transport layer where the relations between the different components of multimedia communications are supported at the transport interface. This allows the specification of a set of powerful QoS parameters and of multimedia synchronisation requirements.

In the multimedia multicast layer, based on the monomedia layer, sessions consisting of multiple monomedia connections can be handled. This allows to

specify the relations and interdependencies between these connections. Moreover, in several communication scenarios, no point-to-point communication but group communication is needed and it is important to maintain the integrity of the group communication over the lifetime of the session.

As a general concept, GCAP uses the partial order, and partial reliability idea. It is based on the consideration that a fully reliable, fully ordered communication increases the communication delay which may be worse than tolerating some loss in multimedia applications. Using a well defined specification, for instance a Petri-Net specification, an application can specify which packets must be received by the application to maintain a given quality. This model can also be used for the synchronization of the media units of a multimedia presentation.

The purpose of the GCAP experiments is to demonstrate the feasibility of the technological approach investigated by GCAP, i.e. of the architecture, the multicast monomedia protocol, the multicast multimedia protocol, and the remote loading and execution capability.

It is important to emphasise that GCAP is defining an architecture that is quite general, and that the protocols and the services designed and built by the project are quite new and innovative; of course, the project will not provide the final fully optimised solution in all cases (remember that the TCP protocol has needed 15 years to be optimised) but the project will provide a complete and coherent framework for easily providing, deploying and evaluating these optimisations later. Of course, the present GCAP solutions will still have to be extended by some follow-up efforts to be made more general and integrated.

This paper first describes the models used to design the protocols and how it has been possible to provide multicast services. The protocols are then introduced and specified. The following section shows the way chosen to deploy the protocol, Active Networking, and the used equipment, the Edge Device. The first experiment results are then given and discussed.

2 The models

In this section we present the multimedia and the multicast models used in the GCAP project for developing the Multimedia Multicast transport protocols .

2.1 The multimedia model

Let us first recall rapidly the global definition of multimedia objects. Then we will address their QoS requirements and their relationship with protocols.

2.1.1 Multimedia concepts

General multimedia applications include a synchronised set of flows of different medias, such as data, voice and video, and it appears that we need to precisely express and understand the multimedia QoS requirements.

In order to fully understand this, a first formal description of multimedia synchronisation was given a few years ago [6] using a set of partial orders, and in important cases using a formal model which is an extension of basic Petri nets.

The Petri net models have precise and simple semantics to handle synchronisation. To overcome time limitations, more general models, also based on extensions of Petri nets, have been proposed for expressing general synchronisation requirements [7]. The main extension deals with the capability to express time intervals and time jitters and their relationship with inter-flow synchronisation. It follows that such models are able to describe all hierarchical general logical and timed multimedia synchronisation.

Note that in such a case a complex multimedia object, i.e. a set of different media and their global synchronisation, can be described by a specific Petri Net based model [5].

It appears quite difficult to handle time today at the Transport level. So we will only consider in this work logical synchronisation at the Transport layer and no time synchronisation. Timed synchronisation will be handled at the user level.

2.1.2 Multimedia and QoS

Also, multimedia applications require different kinds of QoS according to the types of data streams involved. This QoS is at the application level impacted by many parameters, such as reliability, throughput, end-to-end delay, etc.

For example, let us assume that a videoconferencing application has to open 3 different connections : one for transmitting audio, one for transmitting video and one for transmitting signaling.

Typically, the audio stream is a CBR stream (Constant Bit Rate) with an almost perfect reliability, and a very low end-to-end delay to favour a high interactivity between the users. The video stream is a VBR stream (Variable Bit Rate) with a peak and a mean rate, and a given reliability level, because the image definition and the flow requirements depend on the quality required by the user. For instance a very high quality is needed for medical images, a medium to poor quality can be sufficient for tele-teaching, etc.

Finally, the signaling connection has to be fully reliable, and can generally accept a lower throughput and a longer end-to-end-delay than audio and video.

Figure 1 shows the simple illustrative basic model describing a multimedia object with three different QoS.

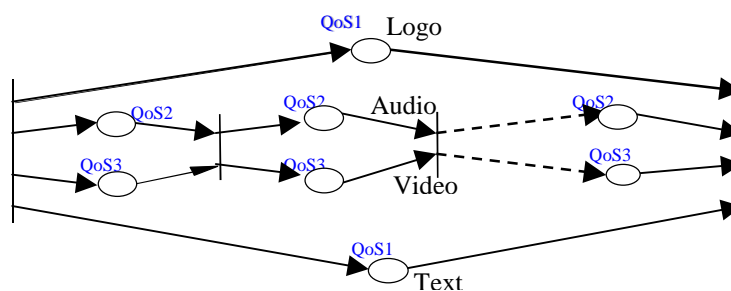


Figure 1. The different QoS of a multimedia object

2.1.3 Multimedia and reliability

On the other hand, it is now well known that the present TCP and UDP Internet transport protocols are not suited for transferring multimedia data streams. This is because they cannot support the various QoS requirements needed by the multimedia applications.

TCP only provides a fully ordered and a fully reliable service, while UDP provides only at the same time an unordered and unreliable service. Multimedia data streams require:

- a partial reliability (in fact it is not troublesome if some images of a 25 images/s video stream is lost or damaged), and
- a partially ordered service (because multimedia applications have to handle several serial and/or parallel compositions of data streams).

The partial order is any order between the total order (as TCP) and no order (as UDP) and it can be expressed as a serial/parallel composition of sub-objects or data units.

As a solution, a partial order transport protocol has been proposed [6] [7] [8]. This new transport protocol that will be extended by GCAP will be called FFTP (Fully programmable Transport Protocol). It will be able to deliver data units sent on one or several connections, following a given partial order, and to have UDP and TCP as two specific cases, as shown in Figure 2.

2.1.4 Partial reliability

Let us now consider the relationship between partial reliability and the QoS of a complex multimedia object. For this, let us use the object given in Figure 1 where 3 different QoSs exist.

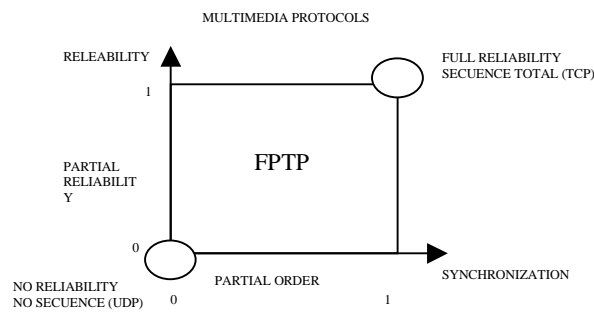


Figure 2 The set of multimedia protocols

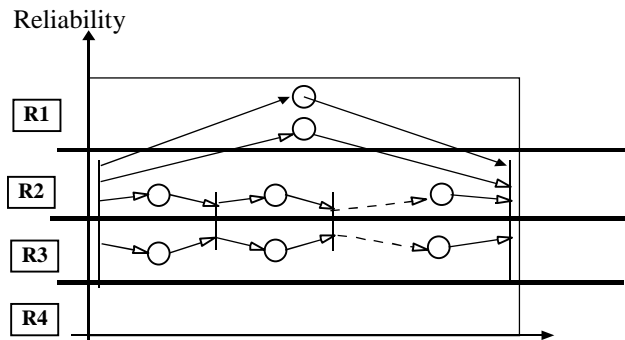


Figure 3. Different reliability levels

Figure 3 now shows the relationship between the QoSs and the partial reliability concept: 3 levels of reliability are needed depending on the sub-objects.

In the case of a videoconferencing system, the order can be seen as the logical synchronisation of the two multimedia streams, i.e. the logical synchronisation of the lip-synchronisation, as this synchronisation is one of the essential key-points of videoconferencing systems.

Furthermore, if we accept the possibility of losses at the user level, i.e. by implicating the possibility of losses in the networks, this leads to the very interesting notion of partial reliability. This notion is tightly related to the communication (network and transport) QoS.

Partial reliability defines a nominal QoS and a minimal QoS under which the user requested service is no more ensured.

Of course, this minimal QoS can be expressed by defining a set of acceptable losses, for instance a maximum number of losses inside a sequence, a maximum number of consecutive losses inside a sequence

2.1.5 Handling losses

When an acceptable loss is detected (i.e. when a received object has a numbering higher than the expected one) the missing object can instantaneously be declared as lost (i.e. leading an earliest indications of losses), and the data (already) received can be delivered at once to the application (i.e. leading to earliest deliveries).

Of course, if the loss cannot be accepted in terms of requested reliability, retransmission will occur. This required partial reliability, defined on top of the partial order model, results in earliest loss indications and deliveries; it is deduced from the application requirements [5].

In fact, two approaches exist for managing partial reliability : media per media and by groups of media. Media per media means that the receiving entity can only handle the partial reliability on the stream it receives and manages, and cannot have any impact on the other streams of the multimedia connection. In the per group of media management, the receiving entity can handle all streams, and so is able to declare losses on the other streams of the same multimedia connection.

2.2 *The multicast model*

The IP Multicast architecture is composed of a service model that defines a group as an open conversation from M sources to N receivers, an addressing scheme based on IP class-D addresses, and routing protocols. In IP Multicast any host can send to a multicast group and any host can join the group and receive data [9]. A multicast group is identified by a class-D IP address which is not related to any topological information, as opposed to the hierarchical unicast addressing model. Therefore, multicast address allocation is complicated and multicast forwarding state is difficult to aggregate. Currently, there is no scalable solution to inter-domain multicast routing.

Hosts express their interest in specific multicast groups through the **IGMP** protocol (*Internet Group Management Protocol*). It enables communication between end stations and their corresponding designated routers. The presently standardized version of the protocol (*IGMPv2* [10]) permits only joining or leaving a group in its entirety.

The first networks implementing multicast (like the *Mbone*) used **DVMRP** (*Distance Vector Multicast Routing Protocol*) [11] as routing protocol. DVMRP is appropriate for a single densely populated domain. It does not scale to the inter-domain level because it is based on flooding techniques to build up a distribution tree for each source. Routers that do not have receivers for a specific group prune the unused branches of the tree.

Other routing protocols were therefore proposed. The presently adopted multicast distribution protocols in the Internet are **PIM-DM** (*Protocol Independent Multicast – Dense Mode*) [12] and **PIM-SM** (*Protocol Independent Multicast – Sparse Mode*) [13]. These protocols build shared trees making the distribution more scalable. PIM-DM is suited for densely populated domains. It is similar to DVMRP, the major difference being that PIM-DM can use any unicast routing protocol, while DVMRP uses its own unicast routing protocol. In the case of sparsely distributed receivers, the recommended routing protocol is PIM-SM. It builds up unidirectional shared trees and can switch to source based trees triggered by the source's traffic rate.

The currently adopted architecture for inter-domain multicast routing in the Internet is based on the **IGMP/PIM-SM/MBGP/MSDP** protocols.

2.2.1 The Tree Building Control Protocol

Tree Building Control Protocol (TBCP) is a new tree building protocol designed at Lancaster University. TBCP can build a loop-free spanning tree connecting TBCP entities that may run either in end systems or in programmable edge devices. Such a spanning tree is built by means of a distributed algorithm that does not require any interaction with multicast routers and any knowledge of network topology.

TBCP has been designed according to the *Application Level Multicast* (ALM) model, also known as application-based distribution. ALM is a new technique used to provide multicast distribution services in situations where no support is provided by the network for multicast. This situation occurs, for instance, when the network infrastructure does not support the IP-multicast routing protocols. Another case in which the ALM approach may be helpful is when the network only supports an SSM multicast service, and the user wants to deploy end-to-end communication protocols that require the availability of a full duplex multicast distribution tree.

When the ALM technique is used to implement multicast communication, each end system that participates in the multicast group is also responsible of forwarding received datagrams to all other ALM entities connected to it in the tree. The advantage of performing data forwarding at the application level is that each end system may be programmed to either passively retransmit every single datagram it receives or actively filter datagrams according to some protocol-specific logic (for

instance, upstream traffic may be aggregated to avoid the source implosion problem, while downstream traffic may be filtered to adapt to the link congestion or the actual computing power of downstream receivers).

In the process of building the TBCP Tree, TBCP entities interact through a sort of signaling protocol which uses point-to-point TCP connections to exchange protocol messages. Once a TBCP Tree has been built, it can be used to deliver UDP datagrams among all the nodes that have joined the tree. For a TBCP Tree to be used by several protocol entities, a set of Data Channels is associated to a TBCP Tree. Each Data Channel is uniquely identified by a 16-bits identifier, called ChannelID. Figure 4 shows a TBCP Tree with three active Data Channels.

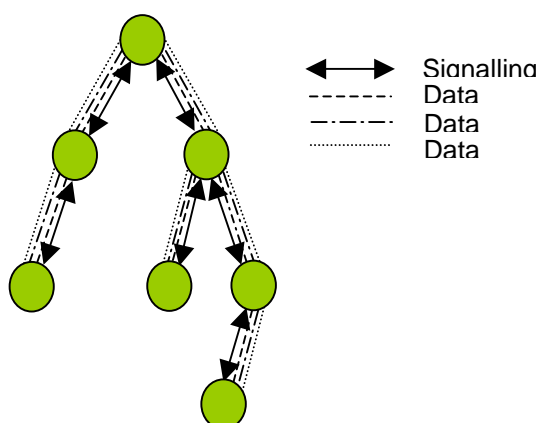


Figure 4. A TBCP Tree with three active Data Channels

The main characteristics of TBCP is the fact that the algorithm used to build the Control Tree assumes only a minimal knowledge of the network topology and partial knowledge of group membership.

3 The protocols

Once explained the model used for specifying the characteristics and requirements of our protocols, we will explain the protocols suit basis developed for the GCAP project in this section and the deployment and experiments done in the following sections.

3.1 The multicast protocols

The suite of protocols that is being defined by the GCAP project is organized in two cooperating planes: Data Plane and Control Plane (Figure 5).

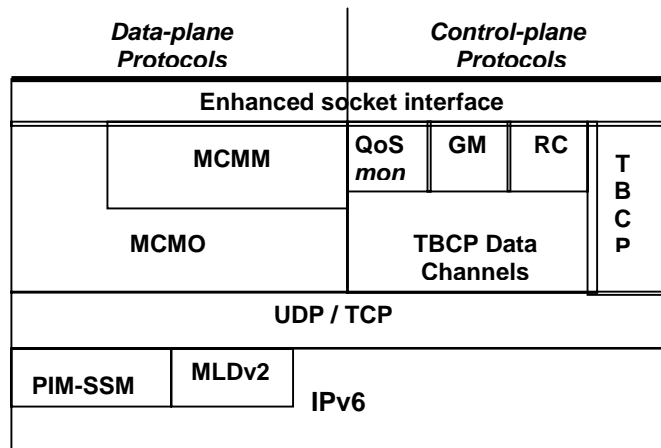


Figure 5. General architecture of the GCAP Protocol Suite.

Data Plane protocols are responsible of end-to-end data delivery to applications. The basic Data Plane communication protocol is the *MultiCast MONomedia* protocol which provides a one-to-many delivery of a single media data units, by relying on a network-layer multicasting service [4]. This protocol provides the following three transport-level services.

- **Fully unreliable service.** This service provides best-effort datagram delivery where no provision is made to enable subsequent attempts to recover lost data.
- **Unreliable with selective recovery requests service.** This service provides best-effort datagram delivery but makes necessary provisions in order to enable the recovery of lost data for a specified recovery period of time. It should be noted that this service is "reactive" in the sense that the recovery period as well as the recovery requests are specified by the service user.
- **Fully reliable service.** This service provides a loss free transfer facility.

On top of the *MultiCast MONomedia* protocol, the *MultiCast MultiMedia* protocol is implemented. This latter protocol relies on the former one to extend the same behaviour to a set of related and synchronized multimedia streams.

Control Plane protocols, on the other side, provide support and management functionality to the applications. To the Control Plane belong the following control protocols, which are currently being defined and specified:

- **QoS monitoring protocol:** provides the application with some clues on the Quality of Service that receivers are currently obtaining. This information can be used by the sender of a data stream to adapt its behaviour.

- **Reliability Protocol:** provides an assisting functionality to help the MultiCast MOnomedia protocol delivery the reliable service that has been requested by the application.
- **Group Management Protocol:** provides the application with a set of primitives to collect information about group members.

All these Control Plane protocols rely on a transport level multicast service provided by the Tree Building Control Protocol, which is described in the next section.

The TBCP protocol supports two different types of operation: TBCP Tree building and maintenance; and Data delivery and reception over the TBCP Tree [4]. These two kinds of operation actually involve two separate (but correlated) parts of a TBCP protocol entity.

3.1.1 TBCP Tree building and maintenance

In this category fall the following activities.

Tree creation / deletion. An application wishing to create a new TBCP Tree rooted at itself instantiates a TBCP_Root entity, and then invokes its `createTree(S, SP)` primitive, where `SP` is the port number used to listen to join requests from other TBCP entities, and `S` is one of the host's IP addresses. Afterwards, to destroy the TBCP Tree, the application invokes the `destroyTree()` primitive on the TBCP_Root entity.

Tree join / leave. When an application wants to join an existing TBCP Tree identified by the couple `(S, SP)`, it instantiates a TBCP_Leaf entity, and then invokes its `join(S, SP)` primitive. As soon as the join primitive is invoked, a distributed algorithm takes place which is aimed at finding the most suitable place for this new entity in the existing TBCP Tree. Such an algorithm is described in the next section of this document ("The TBCP Tree Join procedure"). To leave the TBCP Tree that it has previously joined, an application invokes the `leave()` primitive on the TBCP_Leaf entity that it used to join the tree. An effect of a node leaving a TBCP Tree is that all the node's children entities remain temporarily disconnected. To quickly reconnect these entities to the rest of the Control Tree, a repair procedure is performed.

Tree repair. This activity takes place when a TBCP_Leaf entity detects that it is no longer connected to the rest of the TBCP Tree, and it is aimed at quickly re-establish a valid connection. Hence, its purpose is to keep united the TBCP Tree, even in presence of a receiver crash or loss of connection.

The FFTP protocol has been extended to support the partial reliability service. The partial reliability can be defined in terms of a reliable percentage of ADUs and a maximum value of continuous losses :

FFTPRequest.setReliability(float p) //with $p \in [0,100]$, partial reliability

FFTPRequest.setMaxConsecutiveLosses(int n) //with $n \geq 0$, the maximum number of consecutive losses accepted.

This extension for IPv6 support has been included in the FFTP Socket constructor :

IPv4
ServerFFTPSocket (java.net.InetAddress lAddress, int localPort, int maxConn)
ReceiverFFTPSocket(java.net.InetAddress rAddr, int rPort, java.net.InetAddress lAddr, int lPort)

IPv6
ServerFFTPSocket (java.net. Inet6Address lAddress, int localPort, int maxConn)
ReceiverFFTPSocket(java.net. Inet6Address rAddr, int rPort, java.net. Inet6Address lAddr, int lPort)

The IPv6 extension for FFTP has been developed for the Linux (Linux kernel 2.4.0 and higher) and FreeBSD (3.5.1 and higher).

The multimedia API has been defined, based on a set of monomedia connections and their synchronisation.

3.2.1 The multimedia Application Layer

Now we will explain the services offered by FFTP (Figure 6).

The integration capabilities of FFTP with existent multimedia applications has been demonstrated. In fact, the Java Media Framework player (JMF Player) has been used for the video server application at the sending side and a video player application has been developed at the receiving side.

The FFTP multimedia protocol will be located in the Edge Device systems to allow any multimedia application over IPv4 or IPv6 to communicate using FFTP with specific QoS constraints.

The FFTP protocol will allow a RTP/FFTP translation between the end systems and the edge device. Between the edge devices the QoS constraints will be assured. Note that now, FFTP is both in the end systems and in the Edge Device.

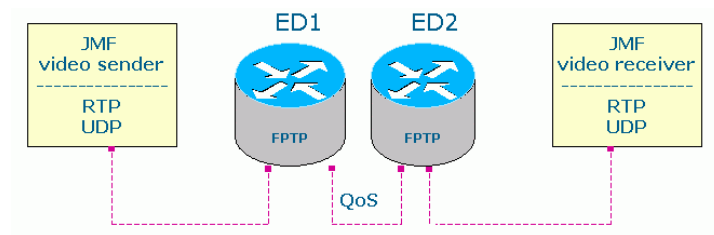


Figure 7. Architecture deployment for the phase III of experiment 1

4 Protocol deployment using Active Networks

4.1 Active Networks

Traditional data networks passively transport bits from one end-system to another. Ideally, the user data is transferred opaquely, i.e., the network is insensitive to the bits it carries and they are transferred between end-systems without modification. The role of computation within such networks is extremely limited, e.g. header processing in packet-switched networks and signaling in connection-oriented networks.

Active Networks break with this tradition by allowing the network to perform customized computation on the user data. For example, a user of an active network could send a customized compression program to a node within the network (e.g. a router) and request that the node executes that program when processing its packets [14].

These networks are “active” in two ways:

- Switches perform computation on the user data flowing through them
- Individuals can inject programs into the network, thereby tailoring the node processing to be user- and application-specific.

An Active Network architecture is formed by two principal components: Active Nodes and Active Packet (Cell).

Active nodes provide a common base functionality. The node architecture deals with how packets are processed and how local resources are managed.

The functionality of the active network node is divided among the Node Operating System (Node OS), the Execution Environment (EEs) and the Active Applications (AAs) [15]. The general organization of these components is shown in Figure 8.

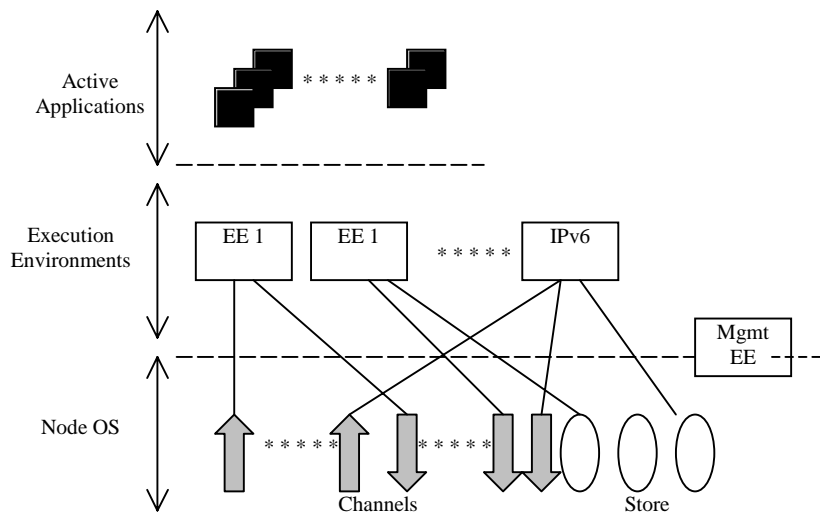


Figure 8. Active Node Components

The NodeOS manages the resources of the active node and mediates the demand for those resources, which include transmission, computing and storage. The NodeOS thus isolates EEs from the details of resource management and from the effects of the other EEs. The EE in turn hide from the OS most of the details of interaction with the end-user [15].

Execution Environments send and receive packets over communication channels.

When a packet is received on a physical link, it's first classified based on information in the packet (headers); this information determines the input channel to which the packet is directed.

On the output side, EEs transmit packets by submitting them to output channels, which include protocol processing as well as output scheduling.

An Execution Environment defines a virtual machine and “programming interface” that can be accessed or controlled by sending the appropriate coded instructions to the EE in packets [15].

An Active Application is a program which, when executed by the VM of a particular EE, implements an end-to-end service. Thus it is the AA that implements customized services for end-user applications, using the programming interface provided by the EE.

4.2 The GCAP active support

It is better to choose for the active networking in the ED a discrete approach instead of a capsule approach.

Thus the Java code downloaded in the IP-Edge Device is loaded from a code repository server to its local code repository with classical transport protocols (ftp, ssh, http, etc).

The following scenario describes the main concepts. We suppose here that the IP ED is empty and contains only a protocol bootstrap.

The execution of the bootstrap may be started with a ssh or rsh ; anyone may be the initiator.

Any IP Edge Device may act as a code repository server for the active mechanisms.

4.3 The GCAP active architecture

Following the model proposed in [16], we can have the protocol functions distributed in end-users and Edge Devices, and we think that protocol loading in the ED can be done in an *Active* way.

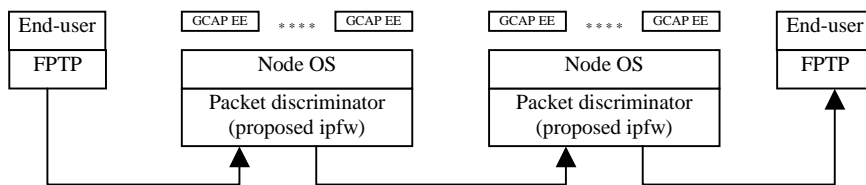


Figure 9. GCAP network active architecture

The communication end-to-end will be divided into shorter connections, for the moment active-node-to-active-node. We can think about the inner nodes as “proxies” for the client-server communication.

There will be a NodeOS in the ED that will load the protocol (EE) dynamically. This EE will be in charge of the next-hop connection till the end-user. The expected architecture is shown in Figure 9.

4.4 An example of an active communication flow

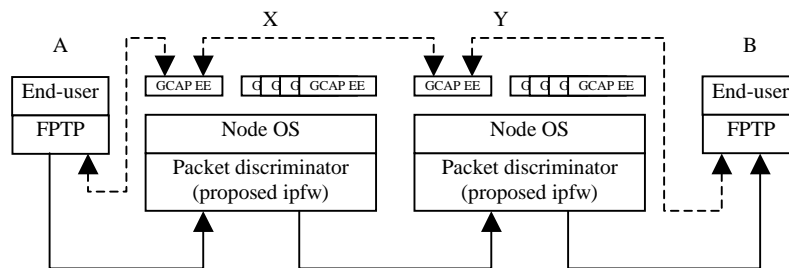


Figure 10. Example of an active communication flow

This is a client-server communication having A as a client and B as a server and X and Y as active routers (Figure 10).

In this figure we can see that the active nodes are going to act as proxies intercepting the communication between two end-systems to convert the local multicast communications to GCAP multicast communications. Each active node will load the GCAP EE (i.e. FPTP and GCAP protocols suit) in a dynamic way, only when needed and will take in charge the communication to the next active node or end-system. The answers and the data and control channels will be managed by the GCAP protocols. This behaviour means that the end-system will start a communication with the other end-system, the active node intercepts the communication and manages it till the end of the communication.

5 The experiments : first implementation results

The proposed application, an advanced videoconferencing system, has to propose to the users to define their respective QoS for each of the exchanged monomedia flows. In this chapter, we describe the videoconferencing system which has been designed, with the architecture of the protocols (point-to-point and multi-point for multimedia), and its integration in the Edge Device. Then we will describe the proposed present tool interface.

5.1 Videoconferencing

On one hand; as already said, this application will provide a configurable support for different QoSs at the transport layer. For each monomedia flow, the users have to define the QoS and the synchronization pattern for the audio and for the video flows. Then these two monomedia flows will be fully synchronized in terms of jitter, for their internal INTRA-flow and for their INTER-flow synchronization.

On the other hand, the set of participants will be managed as a defined group, with its related group control and management. In the proposed demonstration software, the set of the potential participants to the working group will be given before the session. This is why we will integrate a group management module, in which the users will be able to join / leave the working session, and to configure the QoS wished.

5.2 Step one : the point-to-point videoconference

The tests between France and Germany are made by using Internet. We thus launched a point-to-point videoconference over IPv4 using the RTP and UDP protocols.

The formats used are for the video H263 and for the audio G723. So we will present the network captures we made with a network analyser connected just behind the LAAS equipment.

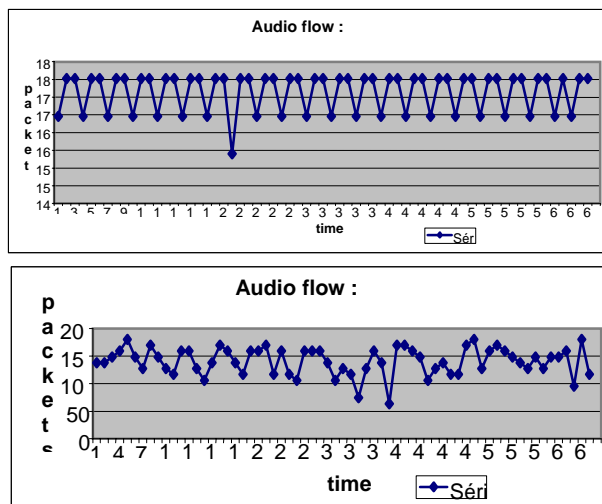


Figure 11. Video flow for the videoconference application

We started by launching an audio conference, then we diffused the video flow only and finally the two flows videoconference was launched. Figure 11 shows the audio flow at the input and at the output of the application. The abscissa represents the time in second, and the ordinates represents the packet sizes. The output flow produced is very regular. This is due to the audio format G723. The audio flow is about 14 Kbps. The figure shows that the audio flow at the videoconference input is an irregular flow, due to the Internet routers. The flow rate is then around 11 Kbps, i.e. leading to about 22% of losses.

The Figure 11 presents the video flow, the input stream and the output stream. The output flow is about 88Kbps with a H263 video format. This rate depends on the video format used by the video capture board.

The input rate is around 68Kbps. This allows us to calculate the loss rate, and our reference is the application output flow. When we compare this flow with the received one, the loss rate is about 22%.

Generally speaking, there was no problem during the diffusion of separate audio and video flows.

Then a different output was obtained later, during the diffusion of both audio and video flows at the same time,

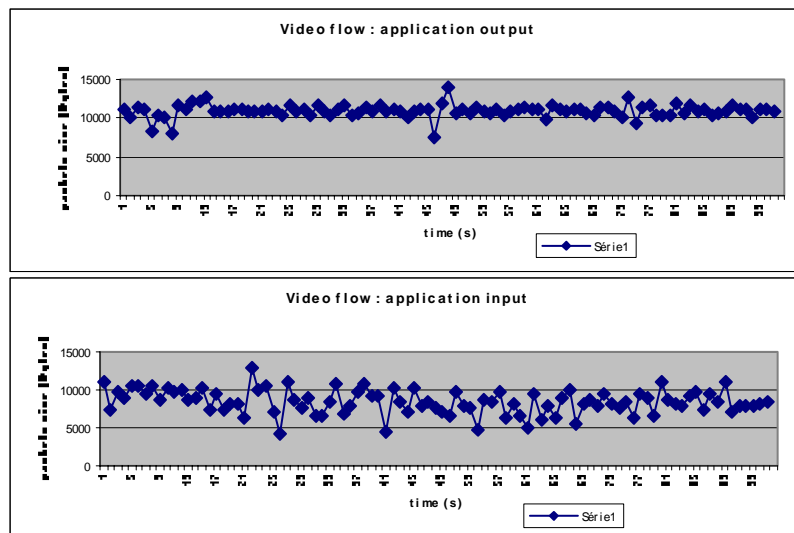


Figure 12. Audio flow for the videoconference application

6 Conclusions and future work

It has been shown in this paper how we have designed and implemented multicast multimedia protocols based on multimedia models. The first results of the implementation have been obtained on a European wide network. These results have shown the feasibility of our protocols, but reduced available bandwidth have introduced delays for complete measurements.

The full experiment is still waiting for homogeneous network structure, and some follow-up work will enhance the solution to support large working groups, maybe up to one hundred users.

References

1. "GCAP: A new Multimedia Multicast Architecture for QoS". Michel Diaz et al.
2. "Specification of the Experimental Platform". M. Diaz, V. Baudin, E. Exposito, D. Garduño, D. Hutchison, D. Larrabeiti, L. Mathy, S. Owezarski, f. Pham-Khac. GCAP report Deliverable Number 4.1.1. July 7, 2001.
3. "Experiment 2 Application Messages". Garduño D, Owezarski S, Diaz M. GCAP report number GCAP-PR-4. June 06, 2001.
4. "Specification of a Multicast Monomedia Protocol". Rolland Vida, Luis Costa, Serge Fdida, University Pierre et Marie Curie, Paris; Roberto Canonico, Laurent Mathy, David Hutchison, Lancaster University, Lancaster. GCAP report number WP2-1. Feb 2001.
5. "Models for enforcing multimedia synchronization in visioconference applications", P. Owezarski, M. Diaz. Proceedings of the 3rd MultiMedia Modeling conference – Towards the information superhighway (MMM96), pp 85-100, World scientific editor, Toulouse, France, November 12-15, 1996
6. "Partial order transport service for multimedia and other applications", P.D. Amer, C. Chassot, T. Connolly, P. Conrad, M. Diaz. IEEE/ACM transactions on Networking, October 1994, Vol. 2 No. 5
7. "From the partial order concept to partial order multimedia connection". C. Chassot, M. Diaz, A. Lozes. Journal of high speed network, Vol.5, n°2, 1996.
8. "The Cesame Project: Formal design of high speed multimedia cooperative systems". M. Diaz, G. Pays. Annals of telecommunications, Tome 49, No. 5-6, May / June 1994.
9. "Host Extensions for IP Multicasting", S. Deering, RFC 1112, August 1989
10. "Internet Group Management Protocol, Version 2", W. Fenner, RFC 2236, November 1997

11. "Distance Vector Multicast Routing Protocol", T. Pusateri, Internet Draft - <draft-ietf-idmr-dvmrp-v3-10.txt>, August 2000
12. "Protocol Independent Multicast Version 2 Dense Mode Specification", S. Deering et al, Internet draft - <draft-ietf-pim-v2-dm-04.txt>, March 2000.
13. "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", D. Estrin et al, RFC 2362, June 1998
14. "Towards an Active Network Architecture". David L. Tennenhouse and David Wetherall. Multimedia Computing and Networking, 1996
15. "Architectural Framework for Active Networks". K.L. Calvert. University of Kentucky; 1999
16. "Design of the Active Protocol entities for the IPV6 support of multi-network ABN". GCAP report 3.1.1
17. "M3POC : a multimedia multicast transport protocol for cooperative applications", T. Gayraud, P. Berthou, P. Owezarski, M. Diaz, 2000 *IEEE International Conference on Multimedia (ICME 2000)*, New York (USA), Jul 30th - Aug 2nd, 2000,
18. "A Time Efficient Architecture for Multimedia Applications", P. Owezarski, M. Diaz, C. Chassot, *IEEE Journal on Selected Areas in Communication*, Special Issue on Protocols and Architectures for Applications of the 21st Century, January 1998.