ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA

CONSIGLIO NAZIONALE DELLE RICERCHE

**S. Belisari, M. Cini, C. Gaibisso, M. Lancia,
M. Vitale**

MOBILE CODE IMPLEMENTATION OF AN AUDIO-CONFERENCE APPLICATION: DESIGN CHOICES AND EVALUATION

R. 486 Novembre 1998

Sabrina Belisari – Tesi di Laurea in Ingegneria Informatica, Facoltà di Ingegneria dell'Università degli Studi di Roma "La Sapienza", svolta presso l'Istituto di Analisi dei Sistemi ed Informatica del CNR, Viale Manzoni 30 - 00185 Roma, Italy.
Email : belisari@iasi.rm.cnr.it.

Marco Cini – Servizio Reti di Comunicazione del CNR, P.le Aldo Moro 7 - 00185 Roma, Italy.
Email : cini@src.cnr.it.

Carlo Gaibisso – Istituto di Analisi dei Sistemi ed Informatica del CNR, Viale Manzoni 30 - 00185 Roma, Italy. Email : gaibisso@iasi.rm.cnr.it.

Maurizio Lancia – Centro Elaborazione Dati del CNR, P.le Aldo Moro 7 - 00185 Roma, Italy.
Email : lancia@iasi.rm.cnr.it.

Maurizio Vitale – Servizio Reti di Comunicazione del CNR, P.le Aldo Moro 7 - 00185 Roma, Italy. Email: vitale@src.cnr.it.

This work has been presented at the "Symposium on Communications Networks and Distributed Systems REDES'98". September 3-4, 1998. Buenos Aires, Argentina

Istituto di Analisi dei Sistemi ed Informatica, CNR
viale Manzoni 30
00185 ROMA, Italy

tel. ++39-06-77161
fax ++39-06-7716461
email: iasi@iasi.rm.cnr.it
URL: http://www.iasi.rm.cnr.it

Abstract

In this paper, the use of a mobile code, partly interpreted language, such as Java, is investigated as a tool to build highly portable implementations of multimedia real time applications. In particular, we concentrate on the implementation of an audio-conference tool. First, we evaluate the efficiency of some Java implemented basic functionality in the real time processing of audio flows. Second, we produce a suitable object oriented design of the tool. Third, the tool has been entirely implemented in Java, with the only exception of some specific audio devices drivers. Finally, we evaluate the tool implementation effectiveness. The obtained results affirm the potentiality of Java in the design and the implementation of real time multimedia applications.

1. Introduction

The great development of Internet technologies and the definition of new programming language environments, such as Java ([1], [2]), meeting the challenges of applications development in heterogeneous, network-wide distributed environments, makes the great diffusion of multimedia applications an extremely realistic prospective. In fact, the major obstacle to the global diffusion of multimedia technologies is that software must be compiled separately to run on different platforms. Java could be the answer to this problem. Programs written in Java can be run whenever the Java platform is present.

This paper investigates the potentiality of the JAVA platform in the design and implementation of real time multimedia applications. We concentrate on the implementation of N-JACT (NetLab Java AudioConference Tool), an audio-conference tool with multicast capabilities developed by NetLab, a research and experimentation laboratory on multimediality and innovative network technologies. A preliminary test activity is finalized to evaluate the efficiency of some Java implemented basic functionality in the real time processing of audio flows. In particular, we focus on the compression/decompression processes, since they are typically the most computationally demanding aspects in this respect. Second, we concentrate on a suitable object oriented design of N-JACT, which makes its main functionality well integrated and easily accessible, thus representing a reference point in the object oriented design and implementation of audio-conference tools. Third, N-JACT has been entirely implemented in Java, with the only exception of some specific audio devices drivers. Finally, we evaluate the tool implementation effectiveness in terms of the user subjective quality of perception and interaction.

Tests have been performed through a framework for the subjective assessment of the quality of audiovisual communications over not guaranteed QoS IP networks, defined and validated by NetLab ([13]). The most significant component of the framework is a network performance modeling system composed by two workstations, connected by an Ethernet LAN (10 Mbps nominal rate), running the multimedia application under investigation. A third workstation hosts a proxy system controlling the network performances. The proxy can be configured in order to introduce packet losses and delays and makes it possible to locally simulate the transmission of multimedia flows between users dispersed in a metropolitan area. Tests have been prepared and performed according to the ITU-T Recommendations defining standard testing methods for the subjective evaluation of the quality of perception ([12]) and interaction ([11]).

Before entering into deeper details, let us briefly illustrate the protocol stack of reference in this paper, shown in figure 1. Main multimedia applications network requirements are low transmission delays and delay jitters, in order to guarantee a good quality of interaction and perception, respectively. The loss of information is a less crucial issue in this context. The adoption of UDP ([8]) is a natural choice in this context, since it guarantees low transmission delays. Unfortunately, UDP still does not keep the delay jitter under control. Consequently, the adoption of a transport protocol, like the Real Time Transport Protocol ([9], [10], [7]), RTP in what follows, especially conceived to improve the steadfastness of the delivered packets interarrival times, is mandatory in order to improve the quality of the user perception. The main idea underlying RTP is the following: as shown in figure 1, let t_i denote the instant in which the RTP source entity makes the i^{th} packet available to the UDP source entity, a_i the instant in which the UDP destination entity makes the i^{th} packet available to the RTP destination entity and p_i the instant in which the RTP destination entity makes the content of the i^{th} packet available to the application. RTP makes it possible to keep the end-to-end packet delay $p_i - t_i$ constant. Such temporal consistence is obtained at the price of an additional buffering delay and of the loss of information. The lower is the accepted delay, the bigger could be the amount of lost information.

A Java implementation of both UDP, provided by the Java Virtual Machine, and RTP, implemented by NetLab ([14]), has been adopted. The potentiality of the Java platform in the design and the implementation of network and transport protocols supporting the real time transmission of multimedia data flows has been already investigated in [14].

2. The Design Activity

The design activity has been mainly finalized to define an object oriented architecture general enough to model the main functionality of a generic audio-conference tool. The software architecture of N-JACT is shown by figure 2. The Java Virtual Machine does not usually directly support hardware devices, like audio and video boards. We consequently implement a C library, and define a Java interface, making it possible for the application to control the input and output audio devices by means of native methods.

The *Network Interface* is mainly in charge of receiving audio samples from the application and to deliver packets to the network. Since N-JACT relies to the services offered by RTP, the *Network Interface* implements the functionality required in order to establish an RTP session, to package together the payload and the header of each RTP packet to be sent and, for each received RTP packet, to extract the payload from the packet and to make it available to the application according to its presentation time.

The *Audio Streams Processing* mainly implements functionality improving the bandwidth utilization. A critical aspect in designing the module has been the choice of the compression/decompression algorithm. Due to the computational restrictions introduced by the Java Virtual Machine, algorithms achieving very high compression rates with a good audio quality, are not suitable to this particular context. ADPCM ([4]) is, as will be shown later, a good compromise among the compression rate, the audio quality and the computational requirements. The *Audio Streams Processing* module has been completed by a *Silence Suppressor*. In order to obtain efficiency and adaptability to the audio signal characteristics we adopt an *end-pointing* algorithm([17]).

The *User Interface* module guarantees a comfortable user-application interaction, making it

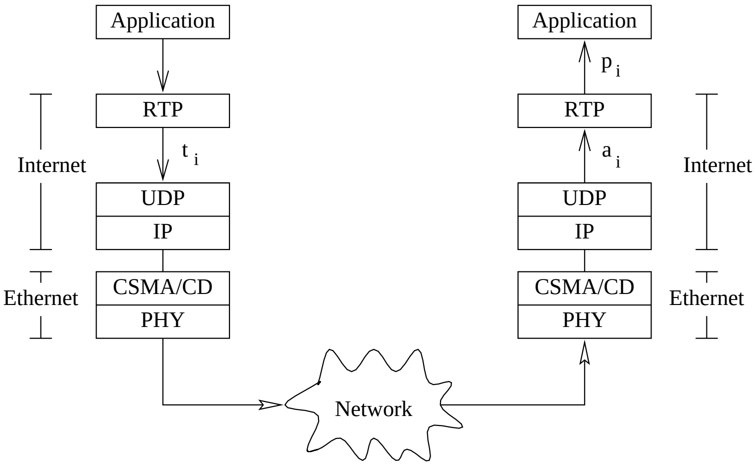


Figure 1: the protocol stack.

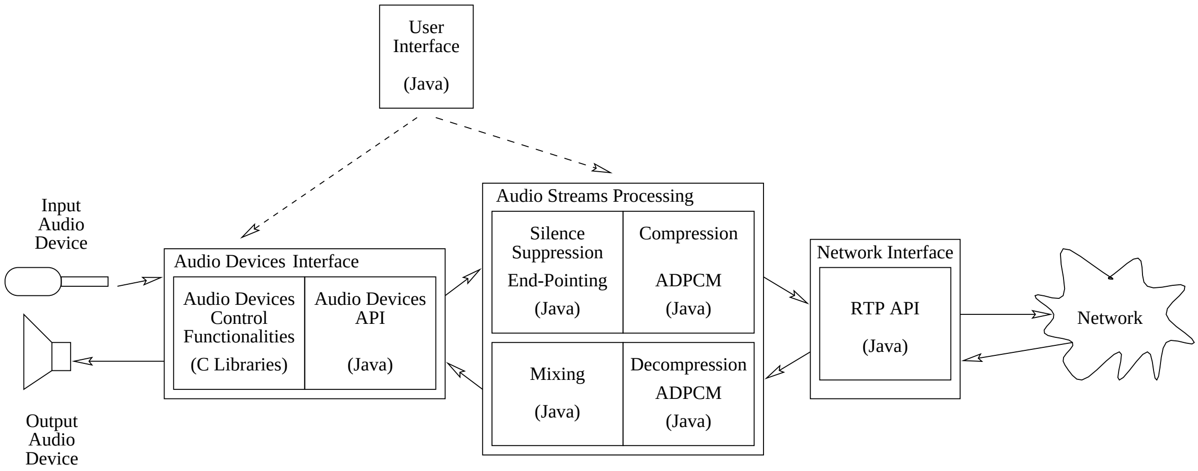


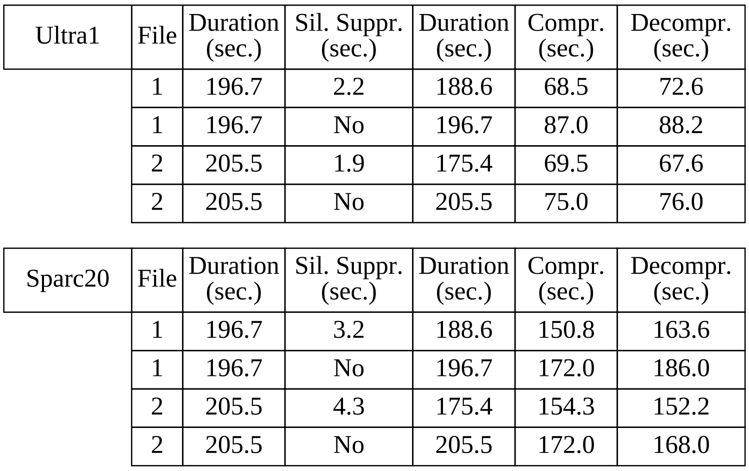
Figure 2: the N-JACT architecture.

possible for the user to enter or to leave an audio-conference session, to control the audio stream processing, as for the example the choice of the compression/decompression algorithm or the compression rate, to manage the audio devices parameters.

3. The Test Activity

A preliminary test activity has been finalized to investigate the efficiency of some Java implemented basic functionality in the real time processing of audio flows. We focused onto the silence suppression and compression/decompression processes, since they are the most critical aspects from this point of view. Several tests have been performed on different hardware/software architectures, processing audio files with different duration and characteristics. For each processed file we measured the original file duration, the time spent by the silence suppression process, the file duration after silence periods have been suppressed, the time spent in order to compress it and, finally, the decompression time. In addition, in order to evaluate the silence suppression impact on the system performance, the time required to compress the original file, and the corresponding decompression time, have been measured. The results of our experiments for two particularly significant files and architectures are shown in figure 3. The impact of the silence suppression process on the performances of the system is quite impressive for both the considered architectures and files. Unfortunately the time spent compressing and decompressing files on the Sun Sparc20 architecture makes the real time transmission of bidirectional audio flows not possible, while the same processes are very efficiently supported by the Sun Ultra1 architecture.

An overall evaluation of the tool implementation effectiveness has been performed by comparing its performances, in terms of the user subjective quality of perception and interaction, with those achievable by IVS ([6]) and Speak Freely ([5]). IVS is one of the first and most widely available public domain software video-conference tool for the Internet. It includes software versions of the PCM ([3]), ADPCM and H.261 ([16]) codecs. The system implements both an error control scheme to handle packet losses and a feedback rate control scheme, which adapts the image coding process to the network conditions. Speak Freely is a Windows audio-conference tool. Like IVS it includes several codecs, PCM, ADPCM, LPC-10 ([15]), all implemented by software. Both IVS and Speak Freely relies to the services offered by RTP.



Ultra1	File	Duration (sec.)	Sil. Suppr. (sec.)	Duration (sec.)	Compr. (sec.)	Decompr. (sec.)
	1	196.7	2.2	188.6	68.5	72.6
	1	196.7	No	196.7	87.0	88.2
	2	205.5	1.9	175.4	69.5	67.6
	2	205.5	No	205.5	75.0	76.0

Sparc20	File	Duration (sec.)	Sil. Suppr. (sec.)	Duration (sec.)	Compr. (sec.)	Decompr. (sec.)
	1	196.7	3.2	188.6	150.8	163.6
	1	196.7	No	196.7	172.0	186.0
	2	205.5	4.3	175.4	154.3	152.2
	2	205.5	No	205.5	172.0	168.0

Figure 3.

The quality of perception has been evaluated according to the ITU-T Recommendation P.85. Different series of tests have been run, characterized by a different percentage of lost IP datagrams. Each test consists in the one-way transmission of several messages. Twenty non-expert subjects have been involved in the experimentation. Each subject has been asked to express his/her opinion on the quality of voice and the level of intelligibility of the messages in terms of the overall impression, listening effort, understanding problems, quality of articulation and level of service acceptance. Figures 4 show the results of our investigation. Each subject classified his/her overall impression, listening effort, understanding problems and perceived quality of articulation as *Excellent*, *Good*, *Fair*, *Poor* and *Unsatisfactory*, to which have been assigned the scores *5*, *4*, *3*, *2* and *1*, respectively. As concerns the level of service acceptance, It has been classified by a *Yes* or a *No*, to which have been assigned the scores *100%* and *0%*, respectively. The opinions expressed with respect to the same indicator have been summarized by the MOS (Mean Opinion Score), defined as the sum of the scores associated to each opinion divided by the number of subjects. As evident, N-JACT better behaves than IVS and its performances are comparable to those of Speak Freely, when the subjective quality of perception is evaluated.

The quality of perception has been evaluated according to the ITU-T Recommendation P.920. Different series of tests have been run, with end-to-end delays distributed according to the Poisson law with mean 0, 300 and 600 msec. Each test is based on a free conversation mainly consisting in the accomplishment of a *name-guessing task*, a question-answer game involving two players. Twenty couples of non-expert subjects participates to the experimentation. Each subject has been asked to express his/her opinion on the quality of interaction in terms of the audio quality, the effort put to interact, communication difficulties and the level of service acceptance.

Figure 5 shows the results of our investigation. IVS has not been considered since it does not support full-duplex communications. Each subject classified the quality of perceived audio and the effort put to interact as *Excellent*, *Good*, *Fair*, *Poor* and *Unsatisfactory*, to which have been assigned the scores *5*, *4*, *3*, *2* and *1*, respectively. As concerns the level of service acceptance and communication difficulties, they have been classified by a *Yes* or a *No*, to which have been assigned the scores *100%* and *0%*, respectively. Once again, the opinions expressed with respect to the same indicator have been summarized by the MOS. The performance of N-JACT are comparable to those of Speak Freely, when the subjective quality of interaction is evaluated.

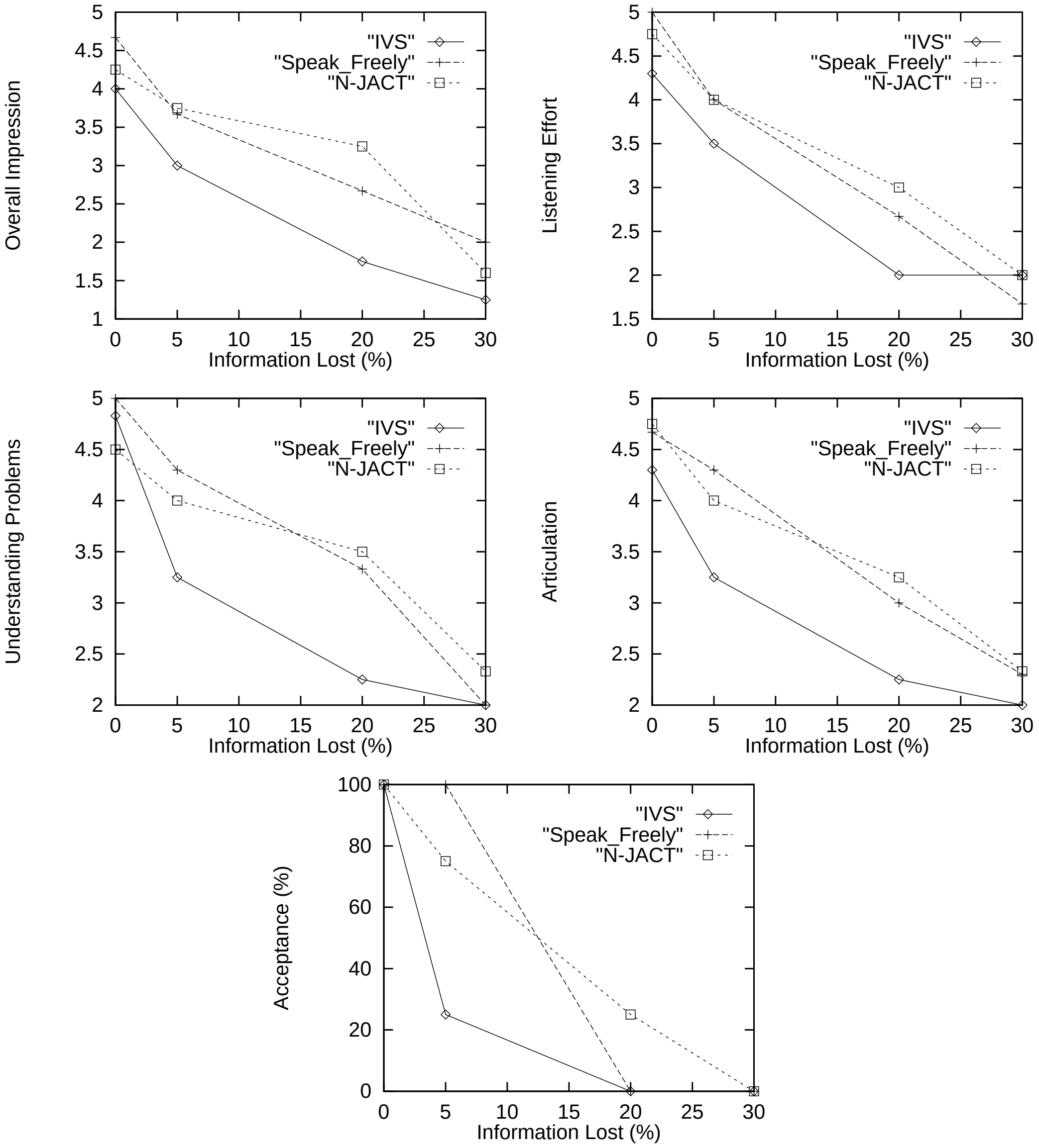


Figure 4.

4. Conclusions

In this paper, the use of a mobile code, partly interpreted language, such as Java, is investigated as a tool to build highly portable implementations of multimedia real time applications. In particular, we concentrated on a Java implementation of an audio-conference tool whose effectiveness has been evaluated both in terms of the user perception and interaction quality. The obtained results have been very encouraging, affirming the potentiality of Java in the design and the implementation of real time multimedia applications.

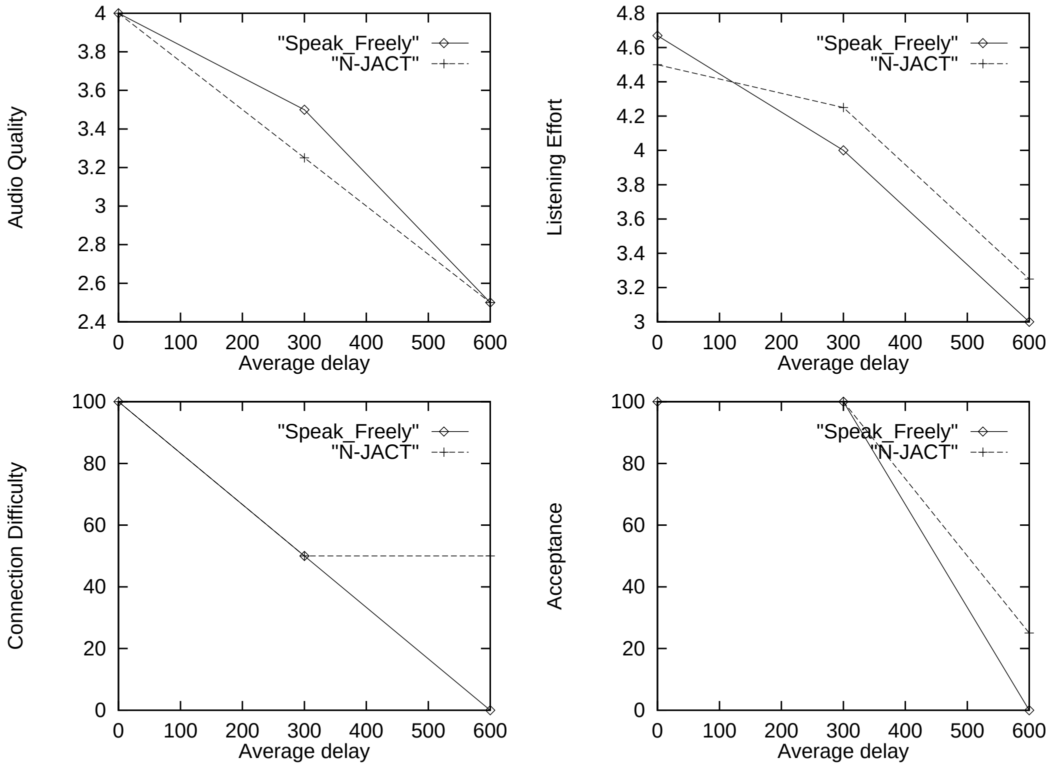


Figure 5.

References

[1] D. Kramer: "The Java Platform, A White Paper". JavaSoft, May (1996). URL: http://www.javasoft.com.

[2] J. Gosling, H. McGilton: "The Java Language Environment. JavaSoft, October '95". URL:http ://www.javasoft.com.

[3] ITU-T: "Pulse Code Modulation (PCM) of Voice Frequencies". Recommendation G.711.

[4] CCITT: "40, 32, 24, 16 Kbit/s Adaptive Differential Pulse Code Modulation (ADPCM) for Audio Encoding", Recommendation G.726.

[5] J. Walker: "Speak Freely for Windows". URL:http://www.fourmilab.ch/speakfree/windows/speak_freely.html.

[6] T. Turletti: "The INRIA Videoconferencing System (IVS)". Connexions - The Interoperability Report Journal, Vol. 8, No 10, Oct. 1994, pp. 20-24.

[7] H. Schulzrinne: "Issues in Designing a Transport Protocol for Audio and Video Conference and Other Multipartecipant Real-Time Applications". Audio Video Transport Working Group, May (1994).

[8] J. Postel: "UDP: User Datagram Protocol". RFC 766.

[9] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson: "RTP: A Transport Protocol for Real-Time Application". RFC 1889, IETF, January (1996).

[10] H. Schulzrinne: "RTP Profile for Audio and Video Conferences with Minimal Control". RFC 1890, IETF, January (1996).

[11] ITU-T: "Interactive Test Methods for Audiovisual Communications". Recommendation P.920.

[12] ITU-T: "A Method for Subjective Performance Assessment of the Quality of Speech Voice Output Devices". Recommendation P.85.

[13] M. Draoli, P. Filosi, C. Gaibisso, M. Lancia, A. Laureti Palma: "A Framework for the Subjective Assessment of Audio Communication on IP Switched Networks: Definition and Validation". To appear.

[14] C. Gaibisso, G. Gambosi, M. Lancia, G. Martufi, E.A. Mastromartino: "Mobile Code Implementation of the RTP Protocol in Java: Design Choices and Evaluation". To appear.

[15] CCITT: "Coding of Speech at 16 Kbits/s Using Low Delay Code Excited Linear Prediction (LP-CELP)". Recommendation G.728.

[16] CCITT: "Video Codec for Audiovisual Services at p*64 Kbits/s Video Coding Standards" Recommendation H.261, Communications of the ACM, April 1991.

[17] "An End-Pointing Algorithm".
fttp://svr-ftp.eng.com.ac.uk/pub/comp.speech/tools/ep.2.0.tar.gz.