The Open Media Streaming Platform

G. Mancini, F. Varano, F. Ridolfo,
E. Masala, J. C. De Martin, A. R. Meo

Internet Media Group
Dipartimento di Automatica e Informatica
Politecnico di Torino

v. 1.0, Jun 27, 2005

Summary

This document describes the Open Media Streaming (OMS) Platform, an innovative client-server open source streaming software suite, fully compliant with the IETF standard. The project started in 2001 from an initiative of the Internet Media Group (http://media.polito.it), a joint research group of the Politecnico di Torino and IEIIT-CNR. The first major, stable release was made in June 2004, when the project was announced to the general public and included in well-know sites, such as Fresh Meat. The Internet Media Group mainly focuses on multimedia coding and transmission, proposing and developing innovative research solutions for multimedia. The OMS software was initially developed as a testbed to experiment with research algorithms in the multimedia transmission field. Then, thanks to the ability of the designers and programmers, OMS evolved in a stable and complete suite of programs that can now be used to provide reliable streaming services in a number of contexts, ranging from audio/video streaming of live events to offline/archive scenarios. Particular attention has been given to the copyright aspects by means of supporting the CreativeCommons licensing scheme (http://www.creativecommons.org), which aims to simplify and standardize automatic handling of copyright issues. These schemes especially foster the cooperation between content producers allowing reuse and sharing of contents on the base of the authors' will. In accordance to these principles, all the source code of the project is freely available on the Internet under the GNU Public License (GPL) at http://streaming.polito.it.

Contents

1 Introduction

The Open Media Streaming Platform (OMS) is a project developed at the Politecnico di Torino and IEIIT-CNR with the aim to build an *open source* streaming platform fully compliant with the IETF standards. The platform is based on the client server paradigm.

The project includes various components:

- the streaming server, called *Fenice*;
- the player, called *Nemesi*;
- a session scheduler, called *Palinsesto*;
- various utilities to support the streaming server.

The Internet Engineering Task Force (IETF), which is the organization responsible, among other things, for the well–known RFC standards, has developed a number of standards concerning multimedia transmission over IP networks. The Open Media Streaming Platform implements the Real–Time Transport Protocol (RTP) [1, 2] for multimedia data transmission, the RTP Control Protocol (RTCP) [1] for the feedback from the client, and the Real–Time Streaming Protocol (RTSP) [3] to manage the multimedia session.

2 Open Media Streaming Architecture

2.1 Server: Fenice

The server can be configured to listen on any specific port for RTSP connections. A configuration file is needed for each multimedia content to stream. In the OMS platform these files present an *.sd* extension. The *.sd* file specifies information such as the name of the compressed file to stream, its encoding format, the payload type to use in the RTP header, and additional information specific for each format, such as frame rate or maximum or preferred packet size.

The algorithm to manage multiple connections is very similar to the one of the well–known Apache web server. Each process can manage a maximum number of streams, specified in the configuration file, after which the server spans a new process. Inside each process, the *select* system call is used to manage the various connections.

The full RTCP protocol is implemented, so detailed statistics for each connection such as mean packet loss rate and delay are available at the server. It is very simple to implement adaptive coding and transmission techniques based on the RTCP information, and the server also offers the possibility to dynamically change the encoding of multimedia data.

The supported players include any RTSP and RTP compliant client. A detailed list is reported in Table 1. Note the compatibility with the RealPlayer, a widespread player installed on most PC. The compression standards supported by the server are listed in Table 2. As shown in the table, the IETF (RFC documents) are implemented for the RTP payload format.

Table 1: Compatibility table: supported players (audio and video).

Player	Platforms	Notes
Nemesi	Linux, Win under devel.	native OMS client
Video LAN Client (VLC)	all platforms supported by VLC (e.g. Linux, Win, MacOS, etc.)	using *live.com* libraries
Mplayer	Linux, Win	using *live.com* libraries
RealPlayer	Linux, Win	v. 8 or above
WinAMP	Win	with *in_rtp.dll* installed

The standard RTSP commands DESCRIBE, SETUP, PLAY, PAUSE, TEARDOWN are implemented. An example of a streaming session showing the RTSP commands is illustrated in Figure 1. The server can be used for both stored and live content. In the latter case, a named pipe is used to supply the server with the compressed video information. In this case, some options are available to mark the source as "live" thus disabling inappropriate RTSP commands such as the "seek" operation. The server can be fed with widely available programs such as *ffmpeg* that are able to grab the video sequence generated by a camera and to compress it in real–time. The video acquisition and compression process and the server can be located in different machines by means of specially–developed utilities, which are described later in this document.

In order to provide a highly scalable architecture, a load balancing mechanism is implemented, by means of the RTSP *redirect* mechanism. This allows to evenly distribute the load on different servers. If different servers for the same content are available, in fact, each server running Fenice is able to redirect the incoming connection of a client to another server if a maximum number of connections has been reached.

The multicast transmission is fundamental when broadcasting the same content to different clients, because useless data replication can be avoided. The Fenice server supports the multicast transmission at IP level. In this case each client can receive the transmission simply joining the multicast group identified by the multicast IP address. As in the "live" scenario, the inappropriate operations such as "seek" are disabled.

Table 2: Compression standards supported by the server.

Standard	Type	Packetization Type
PCM	audio	
MP3	audio	According to RFC 3119 [4]
GSM-AMR [5]	voice	According to RFC 3267 [6]
MPEG-1,2 (ES) [7, 8]	video	According to RFC 2250 [9]
H.264 [10]	video	According to RFC 3984 [11]
MPEG-4 [12]	video	According to RFC 3016 and 3040 [13, 14]

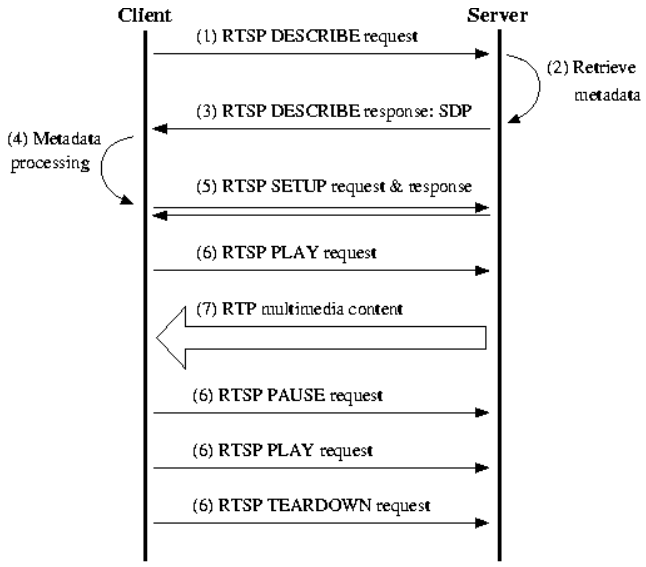


Figure 1: Example of a streaming session showing the RTSP commands.

Utilities are available to manage multicast transmissions, for instance scheduling a certain transmission at a given time and to start and stop streaming on user request. Also the multicast mechanism is implemented as described in the IETF standards, and in particular in Paragraph 14.5 of the RFC 1889 (Real–Time Protocol). For instance, the server carries out an RTSP transaction for each new client that joins the multicast transmission, but no new resources are allocated to it, as specified in the RFC.

The server also implements advanced multimedia processing algorithms, such as the variation of encoding algorithm. The OMS platform in fact allows to dynamically adapt the bitrate of the transmission to the current network congestion conditions, thus improving the quality experienced by the user when the network is in critical conditions. For instance, in the case of audio,the server allows to choose a different audio encoding algorithm for each packet. The encoding algorithm is chosen according to which target bitrate is deemed appropriate for the current network conditions. An adaptive algorithm based on the RTCP receiver reports sent from the client [15] decides at each instant which is the best bitrate to use.

2.2 Player: Nemesi

The Nemesi player is the client natively supported by the OMS platform, thus this is the one that provides the highest compatibility with the server. The player features a multithreaded architecture, in which every logical block is assigned to a different thread. The RTSP protocol management, as well as the RTCP and the RTP protocols, are each managed by a specific thread. Moreover, one thread is dedicated to the graphical user interface. The multimedia decoding process is implemented using a separate thread

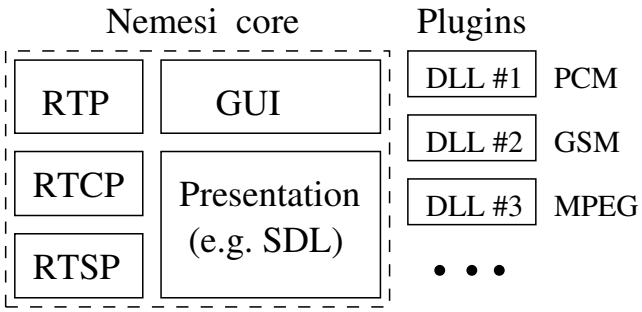


Figure 2: Nemesi architecture. The picture shows the various threads and the plugins for content decoding.

too, as well as the connection with the presentation interface. The SDL library is used as the default interface, but others can be used, for instance, to save the received multimedia data in a file.

A plugin–based architecture is used to allow easy incorporation of any new multimedia coding standard, as shown in Figure 2. Each plugin is implemented as a Dynamic Link Library (DLL) which registers for one or more specific RTP payload types. Thus, each arriving RTP packet is passed by the core of the Nemesi player to the specific DLL that can handle it, on the basis of its RTP payload field. In Linux, the DLL is implemented as a run–time loadable file with *.so* extension.

The implementation of the multimedia decoding engines as DLL loaded at run–time allows the Nemesi source code to be free of any license constraints, thus it can be distributed with the GPL license, even if some DLL's rely on libraries such as the *libavcodec* and *libavformat*, whose inclusion in the project would prevent to distribute it with the GPL license.

Currently Nemesi supports the following formats for audio transmission:

- PCM
- MP3
- GSM-AMR

and these formats for video transmission:

- MPEG-1 and 2 (Elementary Stream; Transport Stream is planned)
- MPEG-4 (almost done)

The graphical interface is based on the GTK+ toolkit library, available for many platforms, including both Linux and Windows. A screenshot of the GUI is shown in Figure 3.

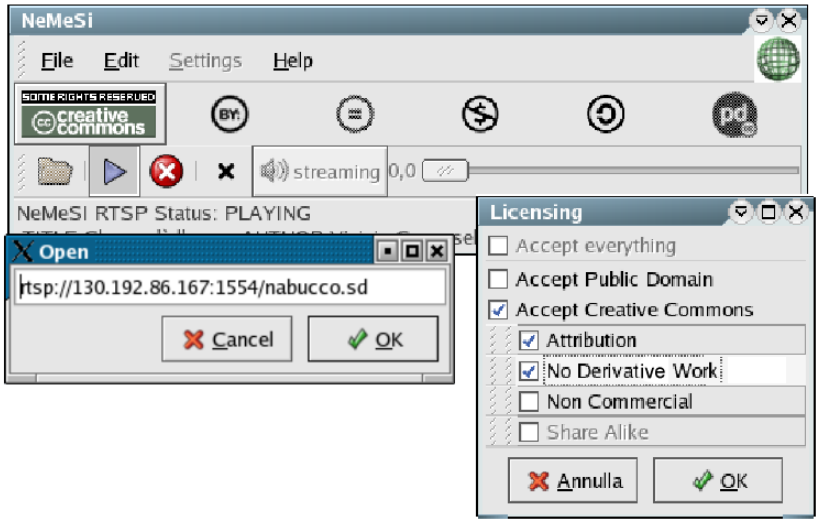
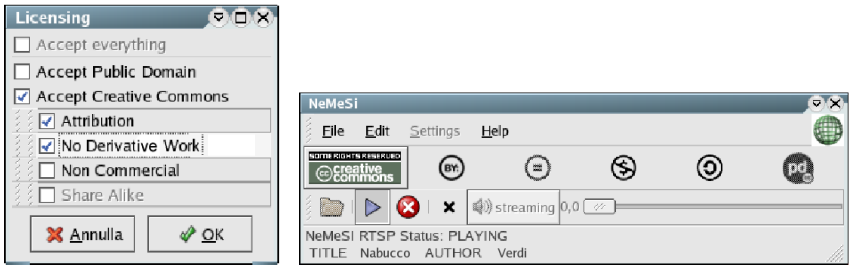


Figure 3: Nemesi screenshot, with the open dialog and the license selection windows.



(a) (b)

Figure 4: CreativeCommons licensing management: dialog box (a) and licensing icons (b).

2.3 Licensing Metadata Support

The OMS platform has been designed to support the exchange of metadata information describing the characteristics of the multimedia contents. For instance, the OMS platform supports the transmission of licensing information according to the CreativeCommons licensing scheme. The CreativeCommons initiative is a project aimed to simplify and standardize automatic handling of copyright issues. These schemes especially foster the cooperation between content producers allowing reuse and sharing of contents on the base of the authors' will. More details can be found at the CreativeCommons official websites [16, 17].

The Nemesi player is able to block or allow reproduction of the received content based on the settings of the client given by the user, as shown in the licence selection window in Figure 4(a). Moreover, it highlights the CreativeCommons licence of the reproduced content using the CreativeCommons standardized icons (see Figure 4(b)). A proposal is currently under development to standardize how the client and the server can communicate CreativeCommons licence information during the RTSP negotiation, by means of an optional attribute. Clients which do not support the licensing information will simply ignore the option. The proposal is also available online [18].

2.4 Session Scheduler: Palinsesto

To manage an Internet web radio or television a scheduler with an appropriate interface is needed to setup and manage the transmissions. The software module of the Open Media Streaming Platform called Palinsesto is a set of PHP pages that facilitates the creation and management of an XML description of the scheduled webcasts. An additional software module, named MClient, loads the XML description and contacts the Fenice server, at the appropriate time, generating an RTSP transaction for each multicast transmission to start.

2.5 OMS Platform Utilities

To provide a really flexible streaming infrastructure, it is necessary to decouple the physical location of the streaming server from the place where the multimedia content is generated. Therefore a mechanism to feed the server with the content to be streamed is needed. With this aim, a software module named Felice has been developed. Its main purpose is to act as a listener for UDP packets on a specific port, and then to repeat them in a named pipe or to send them again towards another Felice module.

This module is fundamental to stream live events. In this scenario a computer, generally a laptop, is located at the event place, while servers are typically located in a computing facility where more computing power and Internet bandwidth is available. The amount of Internet bandwidth is the main constraint that limits the number of connected clients.

Moreover, in many event places, Internet connection might be available but the majority of the networks are protected by firewalls and Network Address Translators (NAT) that prevent incoming connections, thus it is impossible to put a streaming server at those places. Using the Felice software module allows to simply send out a sequence

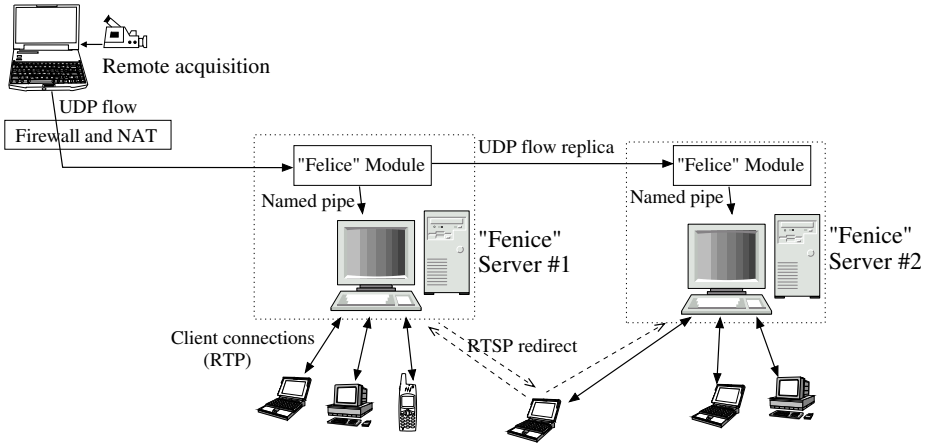


Figure 5: Configuration used for the live streaming of the Internet Governance Working Group (WGIG) meeting.

of UDP packets from the place of the event. This operation is generally allowed by firewalls since the packets come from inside and NAT does not cause any problem to them. An example of this architecture is described in a specific section in this document.

3 Case Study: The WGIG Live Streaming at UN in Geneve

This section describes the streaming architecture that has been used on Feb 15 and 16, 2005 at the United Nations (UN) in Geneve, Switzerland, to provide a live streaming service for the Meeting of the Working Group on Internet Governance (WGIG). Figure 5 illustrates the architecture of the system.

The system could successfully stream the audio/video content of the meeting to about 25 concurrent users. It featured two streaming servers, one in Italy at the Politecnico di Torino, and one in Venezuela. The redirect mechanism was used to balance the load between the two servers. The UN computer network was protected by a firewall, therefore the designed architecture takes advantage of the utility Felice to upload the content from the event remote location to the streaming servers. The Internet router at the UN was configured to guarantee a 384 kbit/s outgoing channel, that was used to stream a QCIF video and two audio channels (English and French). Streaming was carried out using the *ffmpeg* program to compress video and audio and to send it to the Politecnico di Torino using UDP packets. Target bandwidth for audio streams was set at 32 kbit/s (for each language) and video stream bandwidth was about 128 kbit/s. A Felice software module, running at the Politecnico di Torino on the first streaming server, received three UDP streams (one for video and two for audio) and passed them at the Fenice server through three named pipes. At the same time, the same module duplicated the information and sent it over UDP to the second streaming server, lo-

cated in Venezuela, where another instance of the Felice module was running. These module passed the data through three named pipes to the Fenice server, similarly to the Politecnico di Torino case. The content was also saved for later on–demand streaming on the Open Media Streaming Platform streaming server and it is accessible online at http://streaming.polito.it.

4 Acknowledgements

The Open Media Streaming Platform (http://streaming.polito.it) is a work supported by the Internet Media Group (http://media.polito.it) at the Politecnico di Torino and IEIIT-CNR, Italy. The main developers are Giampaolo Mancini, Francesco Varano, Federico Ridolfo and Marco Penno; Juan Carlos De Martin is the lead coordinator and Enrico Masala is the technical counselor.

References

[1] R. F. H. Schulzrinne, S. Casner and V. Jacobson, "RTP: A transport protocol for real-time applications," *RFC 3550*, July 2003.

[2] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," *RFC 3550*, July 2003.

[3] R. L. H. Schulzrinne, A. Rao, "Real time streaming protocol (RTSP)," *RFC 2326*, April 1998.

[4] R. Finlayson, "A more loss-tolerant RTP payload format for MP3 audio," *RFC 3119*, June 2001.

[5] "GSM 06.90 digital cellular telecommunications system (phase 2+); adaptive multi-rate (AMR) speech transcoding," *Draft ETSI EN 301 704*, 1999.

[6] J. Sjoberg, M. Westerlund, A. Lakaniemi, and Q. Xie, "Real-time transport protocol (RTP) payload format and file storage format for the adaptive multi-rate (AMR) and adaptive multi-rate wideband (AMR-WB) audio codecs," *RFC 3267*, June 2002.

[7] ISO/IEC, "MPEG-1 coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mb/s," *ISO/IEC 11172*, 1993.

[8] ——, "MPEG-2 generic coding of moving pictures and associated audio information," *ISO/IEC 13818*, 1996.

[9] R. Hoffman, G. Fernando, V. Goyal, and M. Civanlar, "RTP Payload Format for MPEG1/MPEG2 Video," *RFC 2250*, January 1998.

[10] ITU-T Rec. H.264 & ISO/IEC 14496-10 AVC, "Advanced video coding for generic audiovisual services," *ITU-T*, May 2003.

[11] S. Wenger, M. Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer, "RTP payload format for H.264 video," *RFC 3984*, February 2005.

[12] ISO/IEC, "Information technology - coding of audio-visual objects (MPEG-4)," *ISO/IEC 14496*, 2000.

[13] Y. Kikuchi, T. Nomura, S. Fukunaga, Y. Matsui, and H. Kimata, "RTP Payload Format for MPEG4 Audio/Visual Streams," *RFC 3016*, November 2000.

[14] J. van der Meer, D. Mackie, V. Swaminathan, D. Singer, and P. Gentric, "RTP Payload Format for Transport of MPEG-4 Elementary Streams," *RFC 3640*, November 2003.

[15] G. Davini, D. Quaglia, C. Casetti, and J. C. De Martin, "Perceptually-evaluated loss-delay controlled adaptive transmission of MPEG video over IP," in *Proc. IEEE Int. Conf. on Communications (ICC)*, vol. 1, Anchorage, Alaska, May 2003, pp. 577–581.

[16] "Creative Commons home page," *URL: http://www.creativecommons.org*.

[17] "Creative Commons Italia home page," *URL: http://www.creativecommons.it*.

[18] J. C. De Martin, D. Quaglia, G. Mancini, F. Varano, M. Penno, and F. Ridolfo, "Standard proposal for handling of metadata by IETF-compliant streaming applications – case study: Creative Commons licensing metadata," *URL: http://streaming.polito.it/documentation/ccpl-rtsp*, 2004.