An RTP/SIP Conference Server Based on Linear Mixing Streams

Jorge Crichigno

Technical Report
ECE-440

May - 2006

The University of New Mexico
College of Engineering
School of Electrical and Computer Engineering
Albuquerque, New Mexico
Content

Abstract ................................................................................................................. 2

Introduction ........................................................................................................... 3

Background ........................................................................................................... 4

Design of the Conference Server ........................................................................... 9

Experimental Environment ..................................................................................... 12

Conclusions ........................................................................................................... 13
Abstract

A conference is a conversation among multiple participants. There are many different types of conferences, including loosely coupled conferences, fully distributed multiparty conferences, tightly coupled conferences. In this project, we describe design issues and challenges in implementing a SIP-based tightly coupled conference server. The implemented server meets all the requirements to be used on a SIP-based network. Furthermore, it can be used in the Internet Protocol Multimedia Subsystem, which is the architecture for the convergence for data, speech and mobile networks defined by the Third Generation Partnership Project. Although the project is focused in audio conference, it can be easily extended to video and text conferencing, better known as chatting, which has been growing rapidly over the past few years. This popularity is due to conferencing's ability to simulate a face-to-face meeting in so many realistic ways: for example by enabling file and whiteboard sharing and conveying emotions using video, all in real time.
I - Introduction

In tightly coupled or centralized conferences, there is always a central point of control where each conference participant has a connection. This central point provides a variety of conference services including media mixing, transcoding and participant list notifications [1].

The emerging protocol widely used today and proposed by the Third Generation Partnership Project (3GPP) [1] as the core protocol for the Internet Protocol Multimedia Subsystem (IMS) [1] architecture is The Session Initiation Protocol (SIP) [2]. SIP defines how to establish, maintain and terminate Internet sessions including multimedia conferences. SIP supports various multi-party conferencing models, ranging from mixing in end systems to multicast conferences. When multicast is not available, centralized mixing, transcoding and filtering of media can be used to create multiparty conferences.

In centralized mixing, a server receives media streams from all the participants in a conference, mixes or filters these based on pre-defined policy and distributes the streams to the participants [2]. In audio application, one way to mix streams is the linear mix. In linear mix, streams from different participants are added. Then, the server needs to ensure that a participant does not receive a copy of his own media in the mixed stream. Real Time Transport protocol (RTP) [3] allows a sender to indicate which sources have been combined in a single media packet. When summing, the server should absorb the jitter in packet arrival times while introducing minimum delay (“playout buffer”). For replication, the server should not need to be aware of the media formats. The RTP Synchronization Source Identifier (SSRC) field ensures that the receiver can distinguish different sources
addressed to the same network destination. For either summing or replication, it is desirable if each participant can use different media types and packetization intervals, to accommodate heterogeneity of end systems and access bandwidths. Implementations need to scale to large numbers of conferences as well as large numbers of participants per conference. A media mixing module with a SIP interface can act as a conferencing server component in the distributed application server (AS) component architecture [1]. Advanced system can bundle this functionality with other services, such as interactive voice response (IVR) and a web-based user interface [4].

This project explores the centralized conference server design issues and describes an implementation of such a system.

The remainder of this report is organized as follows. Section II describes the background related to the proposed project. Design and implementation issues of the conference server are presented in Section III. Experimental environments and tests are given in Section IV. Finally, conclusions are presented in Section V.

II - Background

Many PSTN carriers offer conference bridges, which allow users to take part in a voice conference by dialing a telephone number and possibly access code. The same concept can be used for Internet-based conferencing: an arranged conference can be identified by an address, and the participants can join the conference by making a call to this address. One of the most notorious advantages of this approach is that no modifications are required in end systems; similar services like IVR, voicemail and announcements can be added easily by just binding them to addresses. In order to commit this approach, voice over IP (VOIP) protocols disjoint media protocols between media
and signaling protocols. Typically, the protocol used to carry media is the Real-Time Transport Protocol, while for signaling the protocol most widely used today is SIP. SIP identifies the destination address via a SIP Uniform Resource Identifier (SIP URI). Each SIP URI typically has the form sip:username@domain. Some variants are of the form sip:username@ip:port, username@ip (by default the port is set to 5060). The following are examples of SIP URIs: sip: joe@abc.com, sip:moe@216.55.240.91:5060, sip:5052662010@10.20.20.54:3854 [1].

SIP does not define any conferencing entity as such, as these entities are easily modeled as simple user agents. In this model, a conference server is based on the server-based models in which RTP media streams are mixed or filtered by the server and distributed to the participants. There is a standard point-to-point signaling relationship between each participant and the conferencing server. Each relation server-participant relationship is established just like a simple call is established between two users. The conference is identified by a SIP URI, e.g., sip:discuss@server.com. The standard SIP user location and routing mechanisms then forward all calls to the appropriate conference server at server.com without requiring any extension to the protocol. The SIP message routing entities (SIP proxies) need not be aware that the request URI corresponds to a conference. This way, a conference URI can be created by the conference server exactly just like an individual user does, by registering it to the appropriate registrar and publishing it in order to automatically create an ad hoc conference URI globally routable [1].

In order to indicate media type, formats and transport addresses, SIP encapsulates Session Description Protocol (SDP) Packets [6]. SDP is a text-based application-layer
protocol intended to describe multimedia sessions. When describing a session, the caller and callee indicate their respective capabilities, media formats and receive address (IP and port). Capability exchanges can be performed during session setup or during session itself. Generally, the SDP packets are exchanged through the initial SIP INVITE and the subsequent SIP RESPONSE. This scheme follows the SIP offer/answer model [1]. For the conference server presented in this work, when a user wants to be added to the conference session, he sends an INVITE to the conference server. The INVITE message encapsulates the SDP packet describing the capabilities of the user – the offer. This packet contains the set of media streams (one media for audio) and the set of codecs the user wishes to use, as well as the IP address and port the offerer would like to use to receive the media. The conference server replies with the corresponding SIP response message, which encapsulates the appropriate SDP packet. The SDP packet transports the response of the conference server to the previous offer. The answer indicates: 1- whether the media stream is accepted, 2- the codec that will be used in the session, 3- the IP address and port that the conference server will use to receive media [1].

Once established the session between a user and a conference server, the user encapsulates audio samples into frames, which in turn are encapsulated into RTP packets. Then, the latter are sent to the conference server. Figure 1 depicts the Pulse Code Modulation (PCM) process of audio capture, digitalization and framing that must be supported by VOIP phones [7]. The audio sound is captured, digitized, and stored into an audio input buffer 8000 times per second (1 sample each 125 us). This input buffer is commonly made available to the application after a fixed number of samples have been collected. This imposes some delay because the first sample in a frame is not made
available until the last sample has been collected. To avoid an excessive delay, the PCM frame duration commonly used is 20 milliseconds. This is equivalent to 160 audio samples in each frame, which are putting together into the payload of the RTP packet, and then transmitted to the conference server [7].

![PCM Audio capture, digitalization and framing process.](image)

**Figure 1.** PCM Audio capture, digitalization and framing process.

The conference server receives RTP streams sent by each participant. Then, for each RTP packet, the conference server mixes each of the 160 samples contained in each RTP packet payload sent by each participant into one output sample. One RTP output packet is created for each set of 160 output samples, which are put together into the RTP payload. This way, the conference server combines multiple media streams into one, for output.
When mixing, the conference server needs to ensure that a participant does not receive a copy of his own media in the mixed stream [7].

The mixer module at conference server makes use of playout buffers for each arriving media stream to help maintain the timing relationships between streams. At the same time, it has its own Synchronization Source Identifier (SSRC), which is inserted into the data packets it generates. The SSRC is a field in the RTP packet. It uniquely identifies at each participant in a RTP session. The SSRC identifiers from the input data packets are copied into the Contributing Source field (CSRC) list of the output packet. The list of contributing sources (CSRCs) identifies participants who have contributed to an RTP packet but were not responsible for its timing and synchronization [7]. Figure 2 shows the RTP packet format.

Figure 2. RTP packet.
III – Design of the Conference Server

Figure 3 depicts the protocol layer software architecture of the conference Server implemented in this project.

![Diagram of protocol layer software architecture](image)

**Figure 3.** Protocol layer software architecture of the conference Server.

**Media Stack Protocol**

The RTP streams sent by the participants are received and parsed in the RTP layer. This layer provides delivery services for audio frames encapsulated in RTP payload. It also provides payload type identification, sequence numbering, timestamping and SSRC identification. The RTP layer interacts with UDP through sockets, and with the mixer module through a well-defined interface through which RTP layer sends and receives audio frames and associated data to delivery these frames.
**Signaling Stack Protocol**

SIP messages are parsed in the SIP Layer. We have used SIP Express Router (SER) [8] as the SIP layer. SER is a powerful SIP server, which can also be used just as a SIP layer. The SIP layer interacts with the Call Control layer through inter-processes communication, unidirectional named pipes. Figure 4 depicts this interface. When the SIP layer receives a SIP message from a participant of the conference, it parses the message (partially) and sends it to the named pipe server, which in turn sends to the Call Control layer the relevant data. When the Call Control layer wants to send SIP messages to the participants (i.e., when the conference server wants to response to an INVITE message), it passes the messages to the named pipe client using a well-defined interface. Then, the named pipe client sends the messages to the SIP layer. The SIP layer is in charge of the whole SIP concerning issues. These concerning issues include the parsing of the received messages and the management of the transactions with the participants of the conference (message retransmission when necessary, message correlations and timer management of the SIP layer).

![Figure 4. Interface between SIP layer and Call Control.](image-url)
Call Control

This entity makes the decisions of the conference server. It receives SIP messages from the SIP layer and generates the appropriate responses. At the same, it is responsible to keep track the members of the conferences as well as the session information of each of them. Session information includes local and remote RTP transport addresses to send/receive messages. When a participant creates a conference (the first participant in the conference), the call control layer allocates all the necessaries structures for the conference and informs the RTP layer to open the port needed to receive RTP streams from the users. At the same time, it also creates two threads of execution: one for the RTP layer and another for the mixer module. This way, the RTP layer and the mixer module run in separate threads of execution. When the last participant of the conference leaves the conference, the call control layer frees the structures used in the conference and destroys the two threads created at the beginning of the session.

Mixer

Figure 5 depicts the mixer module developed in this project. It shows an example of a conference session with three participants. The conference server just accepts audio encoded with G.711 codecs [9], which is supported by all VOIP phones currently in the market. The mixing algorithm follows the sequence showed in the Figure 5. When an RTP packet (already parsed by the RTP layer) arrives at the mixer module, the samples contained in the payload field are appended to a per-participant audio buffer queue. Each sample is interpreted as an 8-bit linear sample. Generally, the number of samples per RTP
packet is 160, as explained in Section II. Each buffer is labeled with the corresponding RTP timestamp of the RTP packet.

At every outbound packetization interval, a timer triggers a procedure that mixes the samples from one of more input buffers of each active participant. The mixer mixes them into a combined packet by adding the sample values linearly to produce the linear stream X. The timer triggers the mixing procedure at a fixed-time interval of 20 ms. Finally, before to send the mixed stream X to a participant A, the mixer subtracts the A’s input stream from X. This way, the conference server ensures that the participant A does not receive a copy of his own media in the mixed stream.

![Diagram of linear mixer](image)

**Figure 5.** Linear Mixer example: a conference session within three participants.

IV – **Experimental Environment**

The conference server software was development in ANSI C language. The operating system used was Linux Fedora Core I. The source code was compiled with GNU gcc compiler.

The conference server was successfully tested in an environment with several users participating in a conference. Each user was attached to the conference by calling to a
specific number from a X-Lite [10] free available softphone supporting SIP, RTP and G.711 codec. The scenario used for the tests is showed in Figure 6. The devices (users, conference server and SIP server) were attached to a local area network. The user phones were configured to register themselves to the SIP server, which was in charge of the SIP devices in the network. From the point of view of the SIP server and the users, the conference server is just a final user more.

When users dialed a well-known number assigned to the conference server, the SIP server forwarded the messages to it, and the session took place between the users and the conference server.

![Figure 6. Scenario used to test the conference server.](Image)

**V – Conclusions**

This work presents a suitable conference server platform that allows audio conference service without requiring conferencing-aware by the end systems. The implemented server meets all the requirements to be used on every SIP-based network. Furthermore, it can be used in the Internet Protocol Multimedia Subsystem (IMS), which is the
architecture for the convergence for data, speech and mobile networks defined for the Third Generation Partnership Project (3GPP). Although the project focuses in audio conference, it can be easily extended to video and text conferencing, better known as chatting, which has been growing rapidly over the past few years. The conference server supports audio streams encoded with G.711, which is the standard required to every voice over IP phone. The approach used to mix the audio streams from the different users is a linear sum with no weights.

The system was successfully tested in the SIP-based network, where a single SIP Server was in charge of all SIP devices in the network. The tested scenario consisted of a conference session with several users participating of the conference.

As future work, the conference server can be enhanced by considering the following items:

- Adding video support.
- Adding additional audio codecs like G.723, G.729, GSM.
- Adding other services like voicemail, announcement and presence services. These services can be easily added since SIP signaling protocol can be used for each of them. Therefore, the signaling stack protocol of the present project can be used for all of them.
- Improving the mixer algorithm by using a linear weighted sum, automatic gain control of the input audio streams and silence suppression.
- Performing extensive performance testing and evaluating the system.
References


