A Scalable Distributed VoIP Conferencing Using SIP

R Venkatesha Prasad Richard Hurni[#] H S Jamadagni
Centre for Electronics Design and Technology
Indian Institute of Science, Bangalore, India
vprasad@cedt.iisc.ernet.in *richard.hurni@a3.epfl.ch* *hsjam@cedt.iisc.ernet.in*

Abstract

Session Initiation Protocol seems to be the preferred standard for Voice over IP. In this paper, we consider the limitations of existing SIP conferencing methods and propose a distributed architecture using Controllers (SIP Proxy Servers) and Conference Servers which facilitates the control and media handling of a VoIP conference. The Conference Servers are designed on the basis of H.323's Multipoint Processors and handle the audio part of the conference. We provide conferencing messages on SIP that bring these Controllers and Conference Servers together along with the Clients for scalable conference on the Internet. The schematic of the Proxy Server and Conference Server is presented to aid in implementation. We also explain the mechanism we use to improve the quality of mixed output speech by selecting the streams at the Clients.

1. Introduction

The present day Internet uses the TCP/IP protocol suite that uses best effort data delivery. It remains the most widely accepted and supported protocol for Internetworking across all operating systems. Though it was primarily designed and developed for transport of data, it is now used not only to transport data traffic but also for all other types of traffic (like Audio and Video) that have different characteristics and requirements. Therefore, at present, applications like Internet Telephony do not guarantee a minimum acceptable performance. There are a few studies that examine the efficiency and quality of packetized voice [1]: it is found that due to statistical multiplexing and compression schemes, the efficiency in terms of the volume of voice traffic can be increased to a large extent. There are also some studies about the effects of CODEC [23] and QoS guarantees [1, 12].

The next step in the process of merging telephony with the Internet is to offer a number of facilities that a telephone network provides. Among them, the conference facility is the most important as it connects many Clients on a single call, involves mixing their audio streams into a single stream and playing this stream at each Client. In its simplest implementation, the bandwidth requirement for a conference over the Internet is directly proportional to the number of Clients. Thus reducing the bandwidth for conferencing while maintaining audio quality is a real challenge.

In implementing a conference, the issues to be considered are: (a) packet delay, (b) echo, (c) decide the number of Clients that can be speaking simultaneously without degrading voice quality, (d) automatic selection of participating Clients, (e) mix audio from the selected Clients, (f) handle Clients not capable of mixing audio streams ("dumb Clients"), (g) play mixed audio at each Client.

The reasons for popularity of audio or video conferencing on the Internet are dealt with in detail in [24], whereas the advantages of audio and video conferencing have been thoroughly explored in [3] and [4]. This paper is devoted to the techniques of building conferencing applications on the Internet, exploring their architecture and messages. There are two standards for Voice over IP (VoIP) or Internet telephony: ITU-T's H.323 [9] and IETF's Session Initiation Protocol (SIP) [20]. There are many works [25] to consider possible interworking implementations of these standards. Though there are many applications, protocol stacks and installations of H.323 [8, 13], the SIP standard is now in much discussion and seems to be the most sought-after protocol for implementation.

The novelty of this paper is putting all the bits of design information that are proposed in many Internet drafts and the proposed H.323 recommendations for a Multipoint Processor (MP) together to come up with a distributed architecture for an efficient implementation of an audio conferencing system. Distributed, we mean in terms of control messaging as well as audio. Considering only the SIP standard, we propose messages between Proxy Servers, Clients and Conference Servers to make them work together. Though the actual messages structure is beyond the scope of this paper, we explain the steps involved using ladder diagrams and pseudo-messages blocks. We also suggest RTP [22] audio packet format with extended RTP header that is used in our conferencing setup.

The rest of the paper is organized as follows: In section 2, we list the requirements of such a conference facility. In section 3, the ITU-T H.323 recommendations that are useful for implementing the conference facility are explained. Section 4 has a brief description on SIP and limitations of related conference models. In section 5, we detail our proposals along with an example. We finally have the conclusion in section 6.

2. Context and requirements

We address the problem of conferencing under the scheme of distributed "Controllers" and "Media Servers". There are two parts in a VoIP conferencing software. The front-end consists of the "Client" application program that runs on end users' computers. The back-end is provided by other application (server)

[#] Swiss Federal Institute of Technology, Lausanne (EPFL); former visitor at CEDT.

Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC'03)
1530-1346/03 $17.00 © 2003 IEEE

programs that facilitate conferencing. The requirement specifications that are addressed in this paper are:

1. Registration: prospective users (Clients) should first "register" with a control point (like Registrars) to utilize/benefit from the conferencing software.
2. One-to-One call: a Client should be able to call another Client for a one-to-one call.
3. Conference: an ongoing one-to-one call should be upgradeable to a conference by adding a third (and subsequently more) participant(s).
4. Low network traffic: the voice traffic on the network should be as low as possible.
5. Mixing support: in a real-time conference, when more than one participant speaks simultaneously, there is a need to mix audio streams.
6. Consistency: each Client should get the same set of audio streams for mixing.
7. Ease of addition/deletion of Clients: it should be simple to add/delete a Client to/from an ongoing conference.
8. Scalability: the architecture should handle a large number of Clients dispersed over a wide geographical area.

Now we take up a brief explanation of the conferencing architecture as defined in H.323 as our Conference Server, which facilitates the media handling of a VoIP conference, is designed on the basis of H.323's Multipoint Processor.

3. H.323 for audio conferencing

The H.323 recommendation [9] is comprehensive, flexible, and can be applied to a wide range of applications as it defines multimedia communication standards for the IP-based networks.

The H.323 audio conferencing architecture defines a Multipoint Control Unit (MCU) and Clients as the key elements. The MCU is an endpoint in the network, which provides the capability for three or more Clients and Gateways to participate in a multipoint conference. The MCU consists of a mandatory Multipoint Controller (MC) and an optional Multipoint Processor (MP). In a decentralized conference, the Clients talk to each other on multicast or unicast and the MC does not deal directly with the audio streams as the selection of packets and mixing is done at each Client. In centralized conferencing, the MP mixes, switches, and processes audio packets under the control of the MC.

More precisely, the MC performs H.245 [10] multipoint control functions to support conferences between three or more endpoints in a multipoint conference by carrying out the capability exchange between endpoints. It sends a capability set to the endpoints indicating the operating modes in which they may transmit. It may also revise the capability set as a result of Clients joining or leaving the conference, or for other reasons.

The MP receives audio streams from the endpoints involved in a centralized or hybrid multipoint conference, processes these streams and returns them to the endpoints. The MP may process a single media stream or multiple media streams depending on the type of conference supported. An MP that processes audio shall prepare N audio outputs streams from M audio input streams by selecting, mixing, or a combination of these.

The limitation of H.323 is that it does not address the scalability of a conference. The architecture proposes a cascaded and daisy chain topology [11] and it can be easily shown that this architecture cannot scale for a large conference.

4. Session Initiation Protocol (SIP)

SIP is a text-based and HTTP-like application-layer control protocol that can establish, modify and terminate multimedia sessions such as VoIP conferences. The standard defined in IETF's RFC 3261 [20] is a major step in defining a VoIP scenario in the existing Internet, along with RTP [22].

SIP has only three defined entities: User Agent (UA, front-end logical entity generating and receiving SIP requests), Proxy Server (intermediary entity that acts as both a server and a Client for the purpose of making requests on behalf of other Clients, playing a major role in routing and policy enforcement) and Registrar (accepting REGISTER requests and updating the Location Service for the handled domain). The concept of Conference Server is not included in SIP as UAs are capable of doing the work similar to that of an MP in H.323 or a Conference Server can be modeled as a UA. SIP does not offer conference control services such as floor control or voting but it can be used to initiate a session that uses some other conference control protocol.

The core SIP specification supports many models for conferencing [21, 26]. In the server-based models, media streams are mixed by a server whereas in a server-less conference, the mixing is done at the end systems. We shall now consider a few SIP conference models that have been proposed recently [21] along with their limitations. We will use Client (a generic term) and UA (or User) interchangeably in our discussion.

In *End-System Mixing*, one of the UA handles the signaling and media mixing and all other users are in a separate SIP call with this controlling UA, which sends the mixed audio to them. This works for basic conferences and not for a general case with a large number of participants. In the *Users Joining* model, each and every user who invites a new user would transmit the packets that UA receives in the conference, building a growing tree structure but leading to non-scalability due to the increased number of hops. In *Large Scale Multicast Conference*, each UA sends its packets to the group address and receives packets from other UAs on the same address. Assuming that only one participant speaks in a conference and VAD [18] techniques are deployed, this model looks more scalable and easy to implement. The limitations are (i) periodicity of RTCP reports, (ii) security and multicast capabilities of the existing network (especially WANs [21]).

Now considering server-based models, *Dial-In Conference Server* is similar to conference bridges of the traditional PSTN systems. This is the same as the centralized conference model in [26]. The server gets all the streams, mixes them and sends a single mixed stream back to all UAs. As we can see, this model is not scalable as it is limited by the processing power of the server and bandwidth of the network. We do not dwell into *Ad-hoc Centralized Conferences* and *Dial-Out Conference Servers,* as the mechanisms are similar.

Centralized Signaling and Distributed Media: This model has a central controller as in *Dial-In* and *Dial-Out* conferences. But the media is distributed using multicast or multi-unicast (a full mesh architecture as in [26]). Thus for the media, the scalability problems as in the above models persist. The advantage of this model is that the conference control can be a third party solution such as CCCP [5] or SCCP [2].

5. Our approach

We try to have the benefits of many of the above models by proposing to have a distributed control through Proxy Servers. We assume that a large conference involves many users situated in several domains, which are interconnected by a WAN such as the Internet. Looking into these domains in more detail, we distinguish the following conference-related components: (i) an arbitrary number of Clients (UAs), (ii) a SIP Proxy Server coupled with a SIP Registrar Server and (iii) a Conference Server.

5.1. Proxy Server (PS)

In each domain, only one SIP PS is present, coupled with a regular Registrar Server. They can also be two different functional physical entities without altering the scheme in our system. All Clients are connected only to their domain PS or Redirect Server if it is in a different domain.

In addition to its primary role as a means to route requests and responses closer to their final destination, the PS will be able to send triggered INFORM request messages when necessary (see below). Furthermore, it is assumed that all the PSs are stateful, i.e., they remember the state of all calls/conferences under way between one or many Clients in their domain. Thus they will be able to provide the Conference Servers with pertinent information about the sessions. The other part of the entity, namely the Registrar Server, accepts REGISTER requests originating from local Clients and updates the domain Location Service.

Looking at the internal contents of a PS (see Fig. 1), we first note that many interfaces are devoted to communication purposes, either with the Clients inside the given domain, or with other PSs in other domains, and finally with the local CS(s).

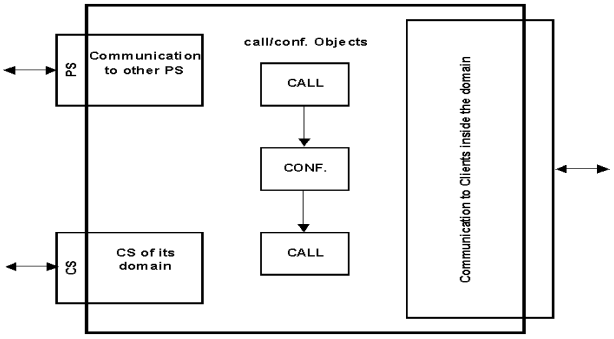

Fig. 1. Internal components of a Proxy Server

Objects regarding the calls or conferences currently under way to/from the domain are stored in the PS so that it has complete information about them (making it stateful). We look at the contents of a Conference Object in Fig. 2: besides the Call-ID of the conference, this object contains other relevant conference-level information (some are not shown here) such as a list of all the Clients taking part in this conference, irrespective of their local or remote location. This list is necessary in case an UA requests a list of the involved parties. A list of the implicated PSs from other domains is also available, which is useful when routing SIP messages across the network. The CS operating in this particular domain is stored (and possibly other CSs if the number of active UAs requires it to ease the load on one CS and to serve many Clients and conferences).

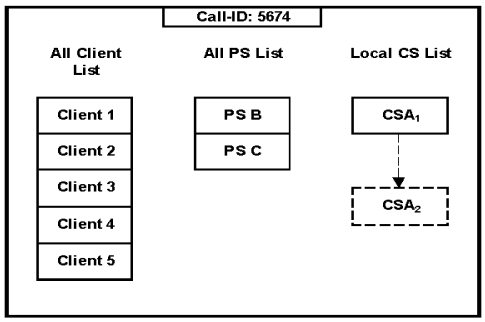

Fig. 2. Detail of a Conference Object contained in a PS

5.2. Conference Server (CS)

A Conference Server (CS) has the function of supporting the conference (and not point-to-point calls) by handling its audio part, comparable to SipConf in [26]. A CS takes the commands from a PS to which it is attached, similar to Selectors of [16]. A functional diagram of a CS is given in Fig. 3a. A CS may be serving many conferences and Clients at an instant of time and many CSs are allowed to exist in a domain.

In the Conference Server, the information about all the CSs involved in the conference is available in a Conference Object (Fig. 3b), along with a flag indicating if the multicast option is enabled or not on that server so that CS-to-CS communication can be on multicast if enabled, otherwise on unicast. This circumvents the usual limitation, the lack of multicast capability in a WAN. When a conference is between many Clients in different domains, many PSs will be involved for that particular conference. The conference state is distributed along with status and other information. We use distributed call state SIP extension for the purpose as in [14]. When we use Silence Suppressions, the packets in a WAN would be not more than one in most cases but, even otherwise, the selection limits it to at most N. Thus the architecture we have proposed here would fit in any scenario and can handle all types of conferences mentioned in section 4.

The role of a CS is to pick N Clients out of M that are in the conference, based on some criterion (e.g. the Loudness Number (LN) [17] of each UA or any other Floor Control methodology). These N (usually three) UAs along with their ID form the set S whose audio packets are to be sent to other CSs handling the same conference. Similarly, for each time slot (packet time), a subset F (which cardinality $|F|$ is fixed to be three [19]) of Clients is selected from the pool of packets from all other CSs and the N selected from the local Clients. The packets in F are received by the Clients from their respective CS, mixed and played out. As CSs are sending the IDs of those N packets, it is easy for the Clients to mix them according to the weights set by the user through a GUI for each of the Clients. This will enhance the quality of the conference as each user can adjust the volume of the speaker to whom he likes to listen. But as the CS sends the subset F to its Clients, some of them may get their own packets back. So while mixing, the weight for the "self" can be set to a very minimal value by the user so that the echo of his own voice is not heard. This also has the advantage that the user knows that his voice packets are selected and heard by all without affecting the quality of the mixed speech at his terminal. Some more facilities can be easily added here: one of the Clients

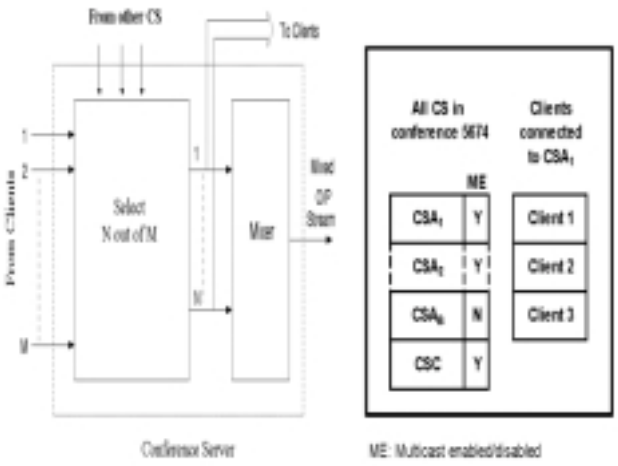

Fig. 3. Conference Server: (a) a schematic diagram and (b) an instance of a Conference Object

can be the moderator and is given priority. We can also give him the responsibility for allowing one of the Clients to be in the subset F. Thus selection of N-1 Clients is needed in this case.

As each CS from a domain will be sending only N out of M packets to the other CSs (M >> N), the bandwidth used by the application over a WAN is bounded above. Each CS will have to handle only N*(P-1) packets (where P is the number of CSs) from other CSs and not M*(P-1). This saves bandwidth and computation at each CS, and leads to a scalable architecture with multiple CSs. Normally, multicasting is allowed inside a LAN, and therefore CS can send multicast packets to its Clients, thus conserving the bandwidth. As the CS chooses the N best Clients out of all the Clients assigned to it in the domain, addition of new Clients to the existing conference will not cause any scalability problem. Furthermore, the audio stream between two Clients will go through at most two CSs in the conference thus fixing maximum of two levels, thus reducing total delay.

Thanks to the use of the LN, the new implementation of the conference facility is close to a real-life face-to-face conference wherein participants try to get (cut) into the conference by force.

5.3. An Example

Now we will introduce a real example to show the capabilities of the proposed architecture. We will assume that we have two domains, namely `mydomain.com` and `remotedomain.com` that are interconnected via the Internet. Both these domains have a PS (coupled with a Registrar Server) as well as one CS. Furthermore, there are two potential SIP UAs in `mydomain.com`, i.e., Alice and Bob, and one in `remotedomain.com`, Ted as shown on Fig. 4.

As seen before, information about the conference resides in the UAs, the PSs and the CSs. All the information regarding the sessions is passed between UAs via the PSs thanks to the SIP Record-Route option. The PSs can take advantage of this communication flow to learn about the session and they can also make use of it to pass information to the recipient by using the attribute line of the SDP [6] payloads. However, SIP as defined in [20] does not provide any request method for communication between the PS themselves or between the PS and the CS. To

Fig. 4. Overview of the example scenario

maintain the SIP philosophy in all communication flows in our conferencing environment, a new request type called INFORM has been created in order for those entities to communicate. This new INFORM request is an ordinary SIP packet that includes a message body containing a SDP message where some line descriptions contain relevant data for the receiving party.

The INFORM requests can be used in different ways:

- A PS in domain *d* can send it to another PS in another domain *d'* to provide it with enough information so that the PS at *d'* has a complete knowledge of the conference (the list of Clients involved irrespective of their location, which are the PS and CS responsible for which of the UAs). For example, when a UA in *d* joins the conference and the PS at *d'* has to be made aware of this, the following session attribute line is added to the session-level description of the SDP payload contained in the SIP packet:
  ```
  a=remote_client_joined:<Client IP address>
    <Handling PS> <Handling CS>
  ```
- A PS in domain *d* will use those INFORM requests to provide its CS with relevant information update. Using the same mechanism explained above, a PS may send an INFORM request with a SDP payload of the form:
  ```
  a=local_client_joined:<Client IP address>
  a=remote_cs_joined:<domain> <Handling CS>
  ```

Similar mechanisms are used when the data about an UA is modified or when it leaves the conference.

We will now have a look at some interesting phases in the conference. We assume that all possible UAs have registered themselves to their local Registrar Server using the REGISTER method so that the Location Service is properly updated. We also assume that all control messages are sent on a different port and may use TCP or UDP (the latter with a timeout).

An interesting step ensues when one of the UA wishes to start the conference. Let us assume that Bob wishes to include Alice (situated in the same domain) in the conference. The flow of messages between Bob and Alice via their PS is a normal session set-up as described in [20]. However, when the "INVITE Alice" request message goes through the PS, the latter adds an attribute line in the SDP description payload of the message to inform Alice of the address of her local handling CS:

```
a=use_local_cs:conference_server.mydomain.com
```

Likewise, when the "200 OK" response comes back from Alice to Bob, the PS adds a similar line so that Bob is also informed.

So far, it was possible to use the possibilities offered by SIP and SDP to convey relevant information to the parties, but we see that we also have to inform the local CS that two UAs have joined after the ACK request is issued by Bob. The sending of an INFORM message from the PS to the CS is thus necessary:

```
INFORM sip:cs.mydomain.com
Via: SIP/2.0/UDP ps.mydomain.com:5060;
     branch=4jfdfg
Max-Forwards: 70
To: <sip:cs.mydomain.com>
From: <sip:ps.mydomain.com>;tag=122678
Call-ID: 265487@mydomain.com
CSeq: 265 INFORM
Contact: <sip:ps.mydomain.com>
Content-Length: 223

v=0
o=bob 20021111142655 20021111142655 IN IP4
  144.16.68.26
s=Audio Conference
c=IN IP4 230.230.230.60/127
t=2873397496 0
a=local_client_joined:144.16.68.26
a=local_client_joined:144.16.68.52
m=audio 8050 RTP/AVP 0
```

The last two lines indicate that two local UAs (with their IP address) have joined the described conference. Now, as both UAs know what CS to use and that the CS knows which UAs it has to handle, the conference can take place.

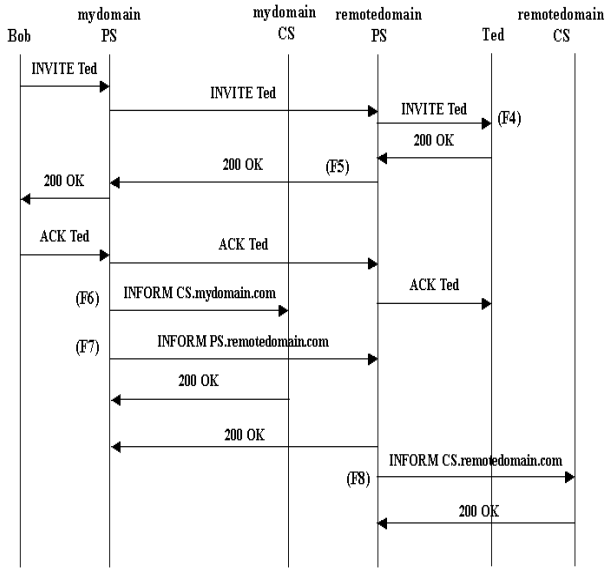

Fig. 5. Message flow for the invitation of a remote UA

The next motivating step happens when one of the Clients in `mydomain.com` (Bob) wishes to include a Client from another domain (Ted). The message exchange is depicted in Fig. 5. In message F4, the PS in `remotedomain.com` informs Ted about his handling CS using the mechanism explained above. When the "200 OK" response comes back from the PS in `remotedomain.com` to the PS in `mydomain.com` (message F5), an attribute line in the SDP payload is added so that it can convey the address of the handling CS used at `remotedomain.com`:

```
a=use_remote_cs:conference_server.remotedomain.
  com
```

This information will be useful when the PS at `mydomain.com` will inform the local CS about that other PS (message F6 shown below in the box with relevant fields only).

```
INFORM sip:cs.mydomain.com
To: <sip:cs.mydomain.com>
From: <sip:ps.mydomain.com>
...
a=remote_cs_joined: remotedomain.com
  cs.remotedomain.com
```

Then, as the PS at `remotedomain.com` was previously not included in the conference, the PS at `mydomain.com` must provide it with some information, in particular about the Clients already included in the conference (F7 partially shown below):

```
INFORM sip:ps.remotedomain.com
To: <sip:ps.remotedomain.com>
From: <sip:ps.mydomain.com>
...
a=remote_client_joined:144.16.68.26
  ps.mydomain.com cs.mydomain.com
a=remote_client_joined:144.16.68.52
  ps.mydomain.com cs.mydomain.com
```

The PS at `remotedomain.com` is now aware of the complete picture of the conference, but it has to provide its CS with some information about it (F8 is shown in the box below):

```
INFORM sip:cs.remotedomain.com
To: <sip:cs.remotedomain.com>
From: <sip:ps.remotedomain.com>
...
a=local_client_joined:203.200.20.95
a=remote_cs_joined:mydomain.com ps.mydomain.com
```

Now all the relevant entities have been provided with the necessary information to make the conference take place between those three parties. Subsequently, whenever a new Client joins the conference, the new PS (if so) would get all the relevant information from the PS that contacts it. All other PSs that are supporting this conference would also be informed by this PS about the new transaction thanks to Conference Objects.

5.4. Audio Data RTP Format

In our case, RTP payload requires some additional header information to be carried in addition to that included in the fixed RTP packet header. The recommended way of conveying this information is in the payload section of the packet [7]. The RTP header extension should not be used to convey payload-specific information [22] since this is inefficient in its use of bandwidth. It requires the definition of a new RTP profile or profile extension and makes it difficult to employ FEC schemes such as [15].

We shall briefly consider the format of the packets, which flow from a UA to the CS and CS to its UAs. The format is given in Fig. 6 and it is an extension of RTP [22]. We have proposed a few extra fields to support new facilities that are accommodated in the payload section of the existing RTP packet format. The packets from the UAs have all the usual RTP fields plus a field for Floor Control Token or maybe Loudness Number. That information forms the extra header (Fig. 6). But the packets from CS to other CS and UAs have sets of Contributing Source ID, Loudness Number and the Audio Data selected. In the example we have assumed that all UAs are using the same payload type.

We may accommodate different payload types by splitting the Floor Control Token field into two for specifying the packet type that fixes the size of the audio part of the packet (not shown). An example of a packet from a CS to its Clients with N=2 is shown in Fig. 6 where the packetization is done for every 40ms. Depending on whether 8 or 16-bit PCM encoding is used, the data length is 320 or 640 bytes respectively.

6. Conclusion

In this paper, we have proposed a distributed architecture with respect to conference control as well as audio data. We have provided a thorough discussion on the PS and proposed the CS and its usefulness in supporting a conference using the SIP standard. The PS and CS are completely distributed therefore the architecture is highly scalable without assuming the availability of multicast support. We have used a conjecture [19] that fixes the number of Clients that are selected in a conference for playing out at each Client. The Clients are fully SIP-compatible and a point-to-point call can be just direct and even without going through PSs if necessary. The messaging between PS and CS is also SIP-compatible and uses only an extended SIP messaging. Moreover this allows the corporate sites to use their own third party CS and other facilities available for conferencing support that is SIP-compatible. The typical RTCP SDES messages, which use appreciable amount of bandwidth can be completely avoided, as the information about the participants is stored in PSs and can be retrieved when required. CSs may act as translators and/or mixers as defined by the RTP [22] standard.

The Clients can use any coding format that is feasible as code translation is done at the CS if required. The selection of $|F|$ Clients globally out of a large number of participants reduces the load on CS when it translates the audio from a Client to another format as at most $|F|$ translations are needed. The security aspects can be taken care of as well because all the audio packets have to go through the CSs and required encryption can be done. Furthermore, allowing only a CS to cross the network boundary can easily accomplish tunneling through a Firewall without compromising the security of the network.

An implementation of the CS on a test bed is fully functional and is working satisfactorily and mimics a natural face-to-face conference. A CS implemented on Intel Pentium-4 1.4 GHz computers running Windows NT can support up to 200 Clients. Our experience with the selection algorithm, the network architecture and protocol between PS and CS is highly motivating for further tests and deployment.

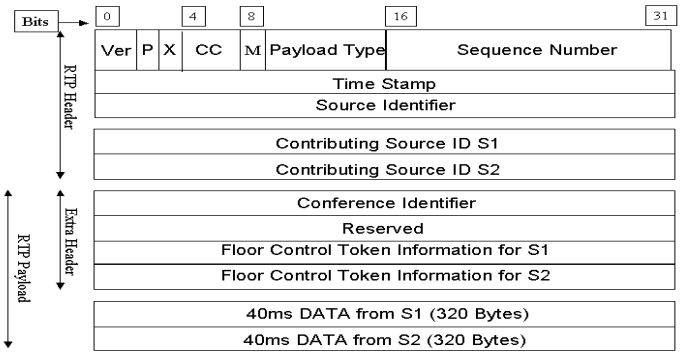

Fig. 6. RTP Packet Structure

7. References

[1] M. Baldi and F. Risso, “Efficiency of Packet Voice with Deterministic Delay”, IEEE Comm. Magazine, May 2000.

[2] C. Bormann, J. Ott and C. Reichert, “Simple Conference Control Protocol”, IETF draft, Dec. 1996.

[3] M. Decina and V. Trecordi, “Voice over Internet Protocol and Human Assisted E-Commerce”, IEEE Comm. Magazine, Sept. 1999.

[4] A. Dutta-Roy, “Virtual Meetings with desktop conferencing”, IEEE Spectrum, July 1998.

[5] M. Handley, I. Wakeman and J. Crowcroft, “The Conference Control Channel Protocol (CCCP): A scalable base for building conference control applications”, Proc. of ACM SIGCOMM 95, Cambridge, MA, pp 275-287.

[6] M. Handley and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, IETF, April 1998.

[7] M. Handley and C. Perkins, “Guidelines for Writers of RTP Payload Format Specifications”, RFC 2736, IETF, Dec. 1999.

[8] Hughes Software Syst., “Voice over Packet H.323 Stack”, www.hssworld.com/voip/stacks/h323/h323.htm

[9] ITU-T Rec. H.323, “Packet based Multimedia Communications Systems”, v. 2, 1998.

[10] ITU-T Rec. H.245, “Control protocol for multimedia communication”.

[11] ITU-T Rec. T.120, “Data protocols for multimedia conferencing”, 1996

[12] M. Korpi and V. Kumar, “Supplementary Services in the H.323 IP Telephony Network”, IEEE Comm. Magazine, July 1999.

[13] S. Kotha, “Deploying H.323 Applications in Cisco Networks”, White paper, Cisco Systems, Inc., 1998.

[14] W. Marshall et al., “SIP Extension for Supporting Distributed Call State”, Internet Draft, Nov. 2000.

[15] C. Perkins, I. Kouvelas et al,"RTP Payload for Redundant Audio Data", RFC 2198, IETF, Sept. 1997.

[16] R Venkatesha Prasad, J. Kuri, H S Jamadagni, H. Dagale and R. Ravindranath, “Control Protocol for VoIP Audio Conferencing Support”, International Conference on Advanced Communication Technology, Mu-Ju, South Korea, Feb 2001, pp. 419-424.

[17] R Venkatesha Prasad, J. Kuri, H S Jamadagni, H. Dagale and R. Ravindranath, “Automatic Addition and Deletion of Clients in VoIP Conferencing”, 6th IEEE Symposium on Computers and Communications, Hammamet, Tunisia, July 2001, pp. 386-390.

[18] R Venkatesha Prasad, H S Jamadagni, Abjijeet et al., “Comparison of Voice Activity Detection Algorithms”, 7th IEEE Symposium on Computers and Communications, Sicily, Italy, July 2002, pp. 530-535.

[19] R Venkatesha Prasad, H S Jamadagni and H N Shankar, “A Conjecture on Specifying Number of Floors in a Voice Only Conference”, accepted for publication in proc. of IEEE ITRE 2003.

[20] J. Rosenberg, H. Schulzrinne et al., “SIP: Session Initiation Protocol”, RFC 3261, IETF, June 2002.

[21] J. Rosenberg, H. Schulzrinne, “Models for Multy Party Conferencing in SIP”, Internet Draft, IETF, July 2002.

[22] H. Schulzrinne et al., “RTP: a transport protocol for real-time applications”, RFC 1889, IETF, Jan. 1996.

[23] R. Shaw, "Reference implementation for CCITT G.711, G.721, G.723", Jun. 1993.

[24] L. R. Silverman, “Coming of Age: Conferencing Solutions Cut Corporate Costs” White Paper, IMCCA.

[25] K. Singh, H. Schulzrinne, "Interworking Between SIP/SDP and H.323". Proc. of the 1st IP-Telephony Workshop (IPTel'2000), April 2000.

[26] K. Singh, G. Nair and H. Schulzrinne, "Centralized Conferencing using SIP". Proc. of the 2nd IP-Telephony Workshop (IPTel'2001), April 2001.