A Proposal for Distributed Conferencing on SIP Using Conference Servers

R. Venkatesha Prasad, Richard Hurni, and H.S. Jamadagni

Centre for Electronics Design and Technology, Indian Institute of Science
Bangalore – 560012, India
{vprasad, hsjam}@cedt.iisc.ernet.in, hurni@ieee.org

Abstract. Session Initiation Protocol (SIP) is the *de facto* standard for Voice over IP implementations. Although this standard does not define conferencing directly, many drafts and papers suggest extensions and solutions to this essential service. In this paper, we first define the issues that are to be considered when building a conference service, and then we motivate our work on some of the limitations of the existing suggested solutions. This framework facilitates the conference control and media handling of a VoIP conference and aims to create a scalable and distributed conferencing system both in terms of load and control. The conference control as well as media is distributed over the network using SIP Servers and Conference Servers, which are described. The floor control is made dynamic by using a metric called Loudness Number [10] that provides a feel of a physical face-to-face conference.

1 Introduction

The growth of Internet encouraged many applications to be ported on the packet-switched network. Transporting the telephony services on the Internet is an exciting assignment and in particular the conferencing facility. We believe that VoIP applications are to be seen from two different aspects: (a) from the technology point of view, and (b) the users' perspective, requirements and interactions.

There are a lot of discussions amongst HCI (Human-Computer Interaction) and CSCW (Computer-Supported Cooperative Work) communities, which mainly deal with the ethnomethodological issues of users' social interactions and about the use of ethnomethodology in designs of applications for CSCW. Unfortunately, the aggressive pace of technical advancements has far outstripped development of metrics and techniques for characterizing and evaluating the novel communication environments. This approach ignores the functional utility of the environment that is used for collaboration [2]. Thus, we take an approach that considers both the aspects, namely, the *technical* and the *functional*.

A conference basically connects many Clients on a single call by mixing their audio streams in the conference into a single stream and playing it at each end-user. For the conference establishment, maintaining and termination the selection of a signalling protocol is necessary. But there are many more issues to be considered when building

A. Marshall and N. Agoulmine: MMNS 2003, LNCS 2839, pp. 32–45, 2003.

a conference: packet delay, echo, deciding the number of Clients that can be speaking simultaneously in a conference without degrading conference quality, automatic selection of Clients to participate in the conference (floor control), mixing of audio from selected Clients, handling Clients not capable of mixing audio streams and finally playing of mixed audio at each Client.

The contribution of this paper includes putting together all the bits of design information that are proposed in many Internet drafts and the proposed H.323 recommendations for a Multipoint Processor (MP) to come up with a scalable and distributed architecture for a SIP conferencing system, both on the conference control as well as the audio media. We introduce Conference Servers that work with SIP Servers and Clients and propose SIP messages for their communication needs. A conference solution using Conference Servers has been tested on a test bed.

2 Problem Formulation

We primarily take voice-only conferences by looking from the telephony service provider's view of the VoIP without considering Video and Whiteboard media. The main argument of our approach is that there is no crucial need for complex floor control methods for voice-only conference. We believe that the conference quality depends not only on the individual speech quality but also on the ability to allow impromptu interruptions. We believe that assuming only one [4] participant speaks at any time often results in losing the interactivity in a conference. There is a recent study by Radenkovic et al. [16], which identifies this need for allowing many speakers to speak at the same instant, supporting our observation of requirement of a conference. They use distributed partial mixing of the speech streams but the drawback is speech losing its spatial aspect when many streams are unnecessarily mixed. Thus we allow very few streams to be mixed only at the time of playing out at the user terminals.

To be precise, we set out with the following wish list for the conferencing service:

1. Prospective Clients should first "register" with a control point to use the conferencing software (allowing for billing if required);
2. An ongoing one-to-one call should be upgradeable to a conference by adding at least a third participant;
3. The voice traffic on the network should be as low as possible;
4. Many participants are allowed to speak simultaneously (only a few of them are selected to speak since allowing everyone does not increase the interactivity; fixing the maximum *number* of simultaneous speakers is dealt in [13]);
5. Each Client should get the same set of audio streams for mixing, ensuring consistency;
6. It should be simple to add/delete a Client to/from an ongoing conference;
7. The scalability must ensure that a large number of Clients dispersed over a wide geographical area are easily handled;
8. Many modes of conference building must be supported (e.g. participants joining the conferences by themselves or by invitation);

9. Conference architecture should also allow the use of features such as floor control without disturbing interactivity;
10. No assumption about the availability of Multicasting in the network.

In this paper we take all the above requirements, some of them implicitly and some explicitly. We use *Users*, *Clients, User Agents* (SIP terminology) and *front end* interchangeably depending on the context.

3 The Motivation

Ramanathan and Rangan [15] have studied in detail the architectural configurations comparing many ways of building the conferencing architecture taking into consideration the network delay and computation requirements for mixing. In [19], a discussion on the architecture for a distributed multimedia conference as well as required control protocol is reported. An overview of many issues involved in supporting a large conference is dealt in [6]. The floor control is another major aspect that should be taken into account while designing/building a conferencing tool and is well documented in [3]. An IETF draft [18] discusses conferencing models with SIP in the background. Implementation for centralized SIP conferencing is reported in [21]. There is a new approach called partial mixing by Radenkovic [16] that allows mixed and non-mixed streams to coexist allowing every ones' speech to be heard. In all the above proposals, while there are some very useful suggestions, they also have the following limitations:

- In an audio conference, streams from all the clients need not be mixed. Actually, mixing many arbitrary streams [16] from the clients degrades the quality of the conference due to the reduction in the volume (spatial aspect of speech). If the number of streams mixed varies, this would lead to fluctuations in the volume of every individual participant causing severe degradation in quality. There is a threshold on the number of simultaneous speakers above which increasing the number of speakers does not improve the conference quality.
- There cannot be many intermediate mixers, because it introduces a lot of delay by increasing the number of hops, therefore it is not interactive.
- The floor control for an audio conference with explicit instructions for participants to talk would result in the conference becoming excessively an artificial one-way speech rather than a live, free to interrupt physical conference.
- Partial mixing [16] has the same problem as that of mixing when more streams are mixed but to allow impromptu speech, mixing is not done when the network can afford high bandwidth requirements for sending/receiving the streams.
- For large conferences [18, 8] a centralized conference cannot scale up. With Multicasting, clients will have to parse many streams and traffic on clients' network increases unnecessarily.

A discussion about the limitations of existing SIP conference models (end-system mixing, users joining, large multicast conference, dial-in conference server, ad-hoc centralized conferences, dial-out conference server, centralized signalling and distributed media) is provided in [12].

4 Our Approach

We will now highlight our approach towards building a scalable and distributed conferencing system. The two parts that must be taken care of in a VoIP conferencing system are the following: (i) the *front-end,* consisting of the application program running on the Clients' computer (end users) and (ii) the other application programs that facilitate conferencing and conference control providing the *back-end*. We will not deal with the first part in detail as we use a regular User Agent as described in [17]. The latter part, however, will be of great interest as it encompasses both the controlling and the media handling sides. In this outlook, we propose to have a distributed control through SIP Servers (SIPSs), whereas Conference Servers (CSs) are special entities that will handle only the audio part of the conference, including floor control.

4.1 Description of the Involved Entities

We consider a large conference with hundreds of participants that are situated in several *domains* distributed geographically and interconnected by a WAN, such as the Internet. A domain can be seen as a LAN with high-speed data transfer, etc. This distributed architecture asks for distributed controlling and media handling solutions, as centralized systems would not scale for such very large conferences. Looking into these domains in more detail (Fig. 1), we distinguish the following relevant entities:

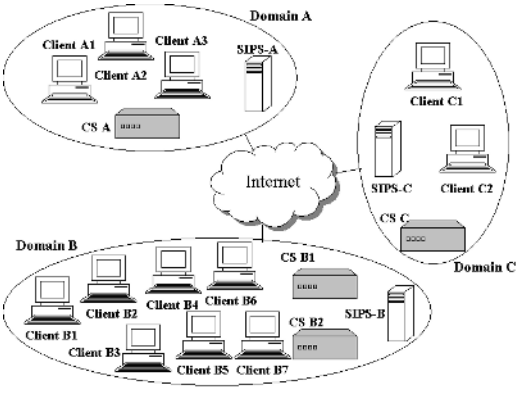

Fig. 1. Description of the different physical components in the proposed architecture.

- An arbitrary number of Clients (User Agents) following the prescriptions in [17].
- A single SIP Server (that will be abbreviated with the acronym SIPS) is present in each domain and is entrusted with all the controlling aspects for the conference(s) under way between one or many Clients in its domain. A SIPS can be viewed as a physical entity that encompasses different logical roles, namely a SIP Proxy Server (which was assumed to be able to generate SIP messages in [12], an assumption that we suppress here because of the presence of the B2BUA), a SIP Registrar Server, a SIP Redirect Server and a SIP B2BUA (Back-to-Back User Agent), which are defined in [17]. This physical implementation enables the handling of in-

coming/outgoing SIP messages by one or another logical entity according to the needs and let the other logical components know about the messages passing through, even if they are not concerned, allowing them to take other actions if necessary. In short, the SIPS can behave like any of the logical roles at any point of time. The B2BUA (and therefore the SIPS) needs to be stateful (in the SIP terminology) and as it is a concatenation of an UAC and UAS, the B2BUA does not need explicit definitions for its behaviour, i.e., it can perform the same tasks and/or handle the same request messages as an UAC or an UAS. The B2BUA, being able to perform more actions than a regular PS, will be the core entity for our conference-related signalling (explained in next subsection).

For the communication needs of the four logical entities in the SIPS, many interfaces are devoted to communication with the domain Clients (User Agents) [12]. Interfaces are also required to communicate with the SIPSs in other domains and with the local Conference Server (see below). Furthermore, objects regarding the calls or conferences currently under way to/from the domain are stored in the SIPS, so that it has complete information about them (fulfilling the stateful condition necessary for the B2BUA as described above). These conference objects (fully described in [12]) contain relevant conference-level information, a list of all the Clients taking part in this conference, irrespective of their local or remote location. A list of the SIPSs from other domains involved in that conference and a list of the CS(s) operating in the local domain are included as well.

The SIPS taking care of all the control aspects has many advantages, viz., (a) it works as a centralized entity that can keep track of the activities of the UAs in a call/conference; (b) it does all the switching for providing PBX features; (c) it locates the UAs and invite them for a conference; (d) billing can be done as well.

- A Conference Server (CS) is a proposed entity that comes into the picture only for the media handling part. Similar to SipConf in [21], a CS has the function of supporting the conference; it is therefore responsible for the audio part, handling audio streams using RTP [20]. A CS only takes commands from its peer SIPS. It can also be used to convert audio stream formats for a given Client if necessary and can work as *Translator/Mixer* of RTP specification behind firewalls. Even if the SIP specification enables direct UA-to-UA media communication in a one-to-one call, it is also possible to use the Conference Server for two-party calls, especially because it is then more functional to create a real conference by adding a third and subsequently more participant(s).

  A CS may be serving many conferences and many Clients at an instant of time in its domain. However, when the number of Clients in a domain is very large, many CSs can coexist in a single domain, each one of them taking care of a fraction of the active Clients. We could implement a CS on a 1.3 GHz Intel PC running Windows NT that supports up to 300 Clients, which shows that even if the number of clients is large, the number of CSs can be kept small.

  In each CS, the information about all other CSs involved in that conference is stored, along with a flag indicating whether its router supports multicast. A list of all the local Clients connected to this particular CS is also available.

We have based the design of our Conference Server on H.323's Multipoint Processor (MP) [7]. In short, the MP receives audio streams from the endpoints, processes these media streams and returns them to the endpoints. An MP that processes audio shall prepare *N* audio outputs from *M* audio input streams by selecting, mixing, or a combination of these. Audio mixing requires decoding the input audio to linear signals, performing a linear combination of the signals and re-encoding the result to the appropriate audio format. The MP may eliminate or attenuate some of the input signals in order to reduce noise and other unwanted signals. The limitation of H.323 is that it does not address the scalability of a conference as the architecture proposes a cascaded or daisy chain topology [8].

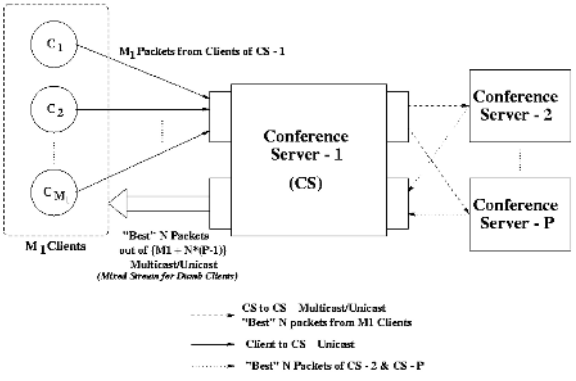

Fig. 2. Schematic diagram of a Conference Server

Continuing with the definition of H.323's MP, the main task of a CS (shown on Fig. 2 and also in [12]) is to pick *N* Clients out of *M* from the local domain that are in the same conference based on some criterion. Audio packets from those *N* Clients, along with their ID are to be sent to the other CSs handling the same conference. Similarly, for each time slot (packet time), a subset *F* of Clients are selected from the pool of packets from all other CSs plus the *N* Clients selected locally and these packets are mixed and played out at the Clients. According to [13], the cardinality of *F*, $|F|$ is *N* and is fixed to three. The flow of audio packets in the system is depicted on Fig. 4 (plain lines).

When comparing our proposal with the ones in the literature, we see that CSs take the commands from a SIPS to which they are attached, similar to Selectors of [9]. However, it makes sense to keep CSs out of all the controlling work from the point of view of a total VoIP service. This is not the case of the Conference Server of [21, 8] which has to do both the work, i.e., conference enabling and floor control.

4.2 Making the Entities Work Together

In order for all the involved entities to work together and provide the desired conferencing service, they need to exchange messages of two different types:

- SIP control messages which are either sent via TCP or UDP, using a timeout in the latter case.
- RTP audio packets, which can be between two UAs in the case of a point-to-point call. Otherwise, first, a Client sends its audio packet to its associated local CS and later CSs exchange their packets (if there are more than one). Then, according to the algorithm defined above, at most three Clients' packets are selected for every time slot and sent on multicast/unicast to other CSs. Then, the CS multicasts the packets back to its associated Clients after selecting the best three streams amongst the streams from other CSs and from its own Clients.

We have already seen that partial or complete information about a conference resides in the User Agents, the SIP Servers and the Conference Servers. To be precise, we draw a list of the conference-level control requirements in the different entities:

- User Agents know the IDs (IP addresses) of their local Register and Proxy Servers (that are fixed and does not have to be dynamically communicated to the UA), which are necessary to start a call or conference. On the other hand, the address of a local CS is not known to the UA in advance, as there can be more than one CS in a domain. There is thus a need to dynamically communicate that address to the UA from the SIPS in its domain.
- A SIP Server knows a lot of information about an ongoing conference: First, it knows the IDs of all registered Clients in a domain and also the IDs of the Clients involved in a given conference. It is also aware of the IP address(es) of the CS(s) handling the media part of the conference(s) in the domain. It also knows the identity of remote SIPS in other domains. Most of the information must be dynamically obtained by the SIPSs when the conference is under way.
- A Conference Server knows the IDs of all the local Clients it must handle the audio packets and the address of remote CSs so that it is able to send a subset of local audio packets in each time slot. As it was decided that CSs would only communicate to their local SIPS for SIP messages, new and dynamic information will have to go through the SIPS before reaching a given CS, and will not go directly between those entities, unlike media packets.

The SIP standard defined in RFC 3261 [17] and in later extensions such as [14] can establish, modify and terminate multimedia sessions. This standard does not offer conference control services, however SIP can be used to initiate a session that uses some other conference control protocol.

For these entities to communicate together and in order to maintain the SIP philosophy in our conferencing system, we assume that the involved entities can both receive and send SIP messages, which is the case of the UAs, the PSs and the B2BUAs that have communication abilities [17]. The only extension needed is to assume that the newly created CSs can send and receive a subset of available SIP requests.

As the different entities need to communicate quite different information between them, we selected a SIP request messages pair that could convey pre-agreed information between these Clients/Servers, namely, SUBSCRIBE and NOTIFY. These two

messages are described thoroughly in [14] and can be shortly described as the SIP framework for event notification where entities in the network subscribe to the state of resources and are notified when those states change. The B2BUA will have a central role, as it will be the SIPS logical component entrusted with handling of all the subscriptions/notifications for all the entities within the SIPS.

In Figure 3, we present a diagram showing the messages pair when used. In message 1, the CS subscribes to be notified of events by the SIPS which message body contains a description of the resource it is subscribing to. This first message is acknowledged by message 2. The SIPS immediately sends a NOTIFY message (3) so that the CS knows the present state of the resource. Later, as soon as an event happens to the subscribed resource, messages 5 and 6 (similar to 3 and 4) are exchanged.

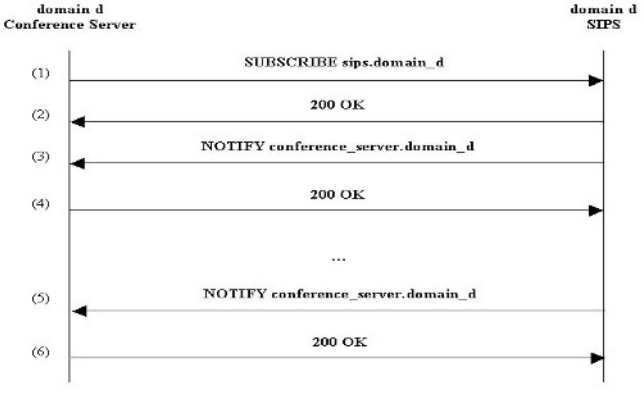

Fig. 3. SUBSCRIBE-NOTIFY message exchange example

In our conferencing environment (and with the help of the defined list of required information every entity is holding), we can draw a precise list of resources a given entity will subscribe to, and also to what other entity it will hold for that particular subscription. The subscription dependencies are drawn on Fig. 4 (dotted lines) and we make the following list as well:

- UAs will subscribe to their local SIPS to be notified to know which CS will handle their audio packets. They may also subscribe to their SIPS to be notified of the joining of every other UAs in order to have a good picture of the conference.
- A SIPS has a subscription with the local registered UAs so that it is informed of changes about their participation in the conference (for e.g. joining, BYE, etc.), making it stateful. They also have to subscribe to the other SIPSs in other domains to pass information to them about the ongoing conference (so it can also provide the local CS(s) with some pertinent information) about UAs joining, and the address of other CSs to pass it on to CSs of its domain. We thus see that by extension all the SIPSs must subscribe with each other (full mesh).
- CSs will subscribe to the local SIPS to be informed about the other CSs present in the conference. It will help in knowing about the UAs, media packets from which it will have to handle in a conference.

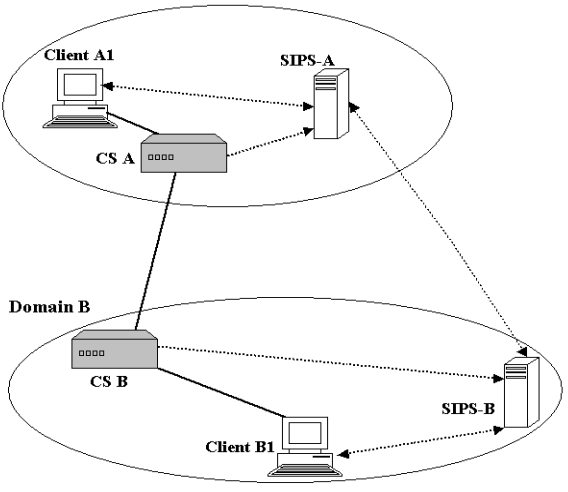

Fig. 4. The message flow between different entities. Signalling messages are shown in dotted lines (with the subscription dependencies), whereas audio packet flow is shown in plain lines.

4.3 Loudness Number (LN)

A basic problem to be solved by the CSs is the following: in a mixing interval, how should it choose *N* packets out of the *M* it might possibly receive? One way to do this would be to rank the *M* packets received according to their energies, and choose the top *N*. This is usually found to be inadequate because random fluctuations in packet energies can lead to poor audio quality. This indicates the need for a metric different from mere individual packet energies that should have the following characteristics:

- A person who is speaking should not be easily cut off by transient spikes in amplitude of the other participants. This implies that a speaker should have some "weight" depending on his past activity (also called "persistence" or "hangover").
- By the same token, a participant who wants to interrupt the speaker will have to raise his voice and keep trying for a little while in order to break in. In a real-life conference, the body language of a participant often indicates that he wants to interrupt. But in the audio conferencing scenario under discussion, a participant's intention to interrupt can only be conveyed through the loudness metric on the basis of which the packets to be mixed are selected.

As a result, we use LN for conference control and floor control. A CS selects at most three Clients to speak to the participants of a conference. The LN and its generation are discussed in [10] and as such we have omitted a thorough discussion about it here.

4.4 Hybrid Floor Control

We propose two floor control mechanisms: *(1) Distributed Floor Control* that preserves the interactivity by means of using LN with three floors. The interactivity is

preserved due to the design of LN and allowing all the participants to speak impromptu to the participants. *(2) Compensatory Floor Control* is used for avoiding the conference becoming messy, which is a remote possibility. For the purpose, we allow one reserved floor apart from the freely available three. A distributed mutual exclusion algorithm suggested in literature [4] running at PSs may be used for controlling this floor; besides, we propose to grant this floor to the conference moderator (if any, and if required). These two floor controls co-exist since the mixing is done at the clients (or last CSs if there are some dumb clients).

4.5 An Example

We will now introduce a real example of the capabilities of the proposed architecture. We assume the following configuration: two domains, namely `Domain A` and `Domain B` that are interconnected by the Internet. Both those domains have a SIP Server (`SIPS-A` and `SIPS-B`) as well as one Conference Server (`CS A` and `CS B`). Furthermore, there are two potential SIP Clients in `Domain A`, i.e., `Client A1` and `Client A2`, and one in `Domain B`, `Client B1`.

Intra-domain setup phase. We will first introduce the phase that occurs in each individual domain before any conference can take place and is necessary so that all involved entities perform some pre-conference required tasks. The message exchange for domain A is shown on fig. 5, where we can distinguish into 3 different parts:

- Part 1 sees the registration of `CS A` to `SIPS-A`: That way, `CS A` will be informed about UAs joining on one hand and about the addresses of other CSs it will have to send media packets to, on the other hand. The "`NOTIFY CS A`" request contains no information but is compliant with the SIP specification.
- Part 2 and Part 3 consist of each Client registering with its local Registrar Server. Then, the clients subscribe to their local SIPS so that they are kept informed of future calls/conferences and CS handling them. After that, `SIPS-A` registers to the client to be kept informed about its state and leading to a stateful SIPS.

After this initial setup phase, a conference may start at any time inside a domain.

Inter-domain setup phase. After the intra-domain setup phase and before any conference can take place between different domains, an inter-domain setup phase must take place. This phase would actually be contained in a “remote invitation phase” (explained below). When the SIPS at `Domain A` receives the third message of the INVITE/200/ACK three-way handshake used to establish SIP sessions, it is then sure that the remote Client situated in `Domain B` will be included in the conference and thus `SIPS-A` subscribes to `SIPS-B` so that it is kept informed about the events in their respective domains. In part 2, the same occurs in the opposite way. Then, if many more domains are involved in the conference (e.g. `Domain C`), `SIPS-A` will make sure that those other SIPSs know about the joining of the domain:

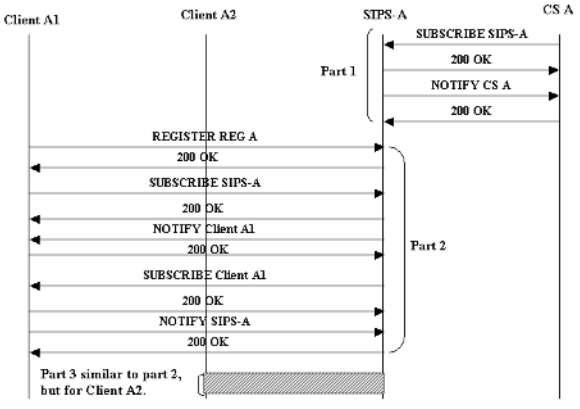

Fig. 5. SIP messaging during intra-domain setup phase

- SIPS-A tells SIPS-B about SIPS-*j* (and all other SIPSs if any). Then SIPS-B would contact each and those would also subscribe to SIPS-B. Whenever SIPS-B subscribes to SIPS-*j*, it would be reciprocated by SIPS-*j* subscribing to SIPS-B.
- In a conference where users join themselves (also called an open conference), then the SDP [5] would have the initiator domain (say SIPS-*j*) to which the new SIPS-*k* would subscribe and get the information about all the other SIPSs.

However, only one inter-domain setup phase can occur between SIPSs in each conference as after this phase, they will not need to start another setup phase as they can already communicate. When all SIPSs in a conference have subscribed to each other, a full mesh of SIPSs is created, which is a guarantee that signalling information is passed according to the needs. In part 3, each SIPS notify their local CS about the ID of the remote CS that they have just been notified of in the form of NOTIFY message.

Local Invitation. Now, we will assume that `Client A1` invites `Client A2`, which are in the same domain. This will start the conference (actually a one-to-one call).
In the first part, `Client A1` invites `Client A2` with a normal SIP invitation messages exchange as defined in [17]. The messages go through the local Proxy Server, which is contained in `SIPS-A`. In the second part, `SIPS-A` sends a NOTIFY message to `CS A` to inform it that two local Clients (with their IDs) have joined the conference. That way, `CS A` will be able to handle their audio packets. In the third part, `SIPS-A` notifies `Client A1` and `Client A2` with the ID of `CS A` so that they will know which is their media handling Conference Server.

Remote Invitation. The next motivating step happens when one of the Clients in `Domain A` wishes to invite a Client from `Domain B`. Let us assume that `Client A1` invites `Client B1` in the conference. The message exchange for remote invitation is depicted in Fig. 6.

In part 1, a normal SIP invitation takes place between `Client A1` and `Client B1`. But as the ACK message flows through `SIPS-A`, and if the inter-domain setup has not already occurred between the two involved domains, this phase must take place right away. In the future, this will not happen again as this is an only-once phase. In part 2, `SIPS-A` notifies `SIPS-B` about relevant conference-level information (such as the handling CS in the domain, the ID of the involved local Clients). Then `SIPS-B` does the same with `SIPS-A`. Then, in part 3, `CS A` is notified of the ID of `CS B` that has just joined the conference (known through the previous messages exchanged). In part 4, `SIPS-B` does the same with its local Conference Server, `CS B`.
Now that all the entities have been provided with the necessary information to make the conference take place between the involved parties, the conference can start. The deletion of clients is done according to the same scheme when necessary, i.e. sending of normal BYE SIP messages between entities and notification of the other parties.

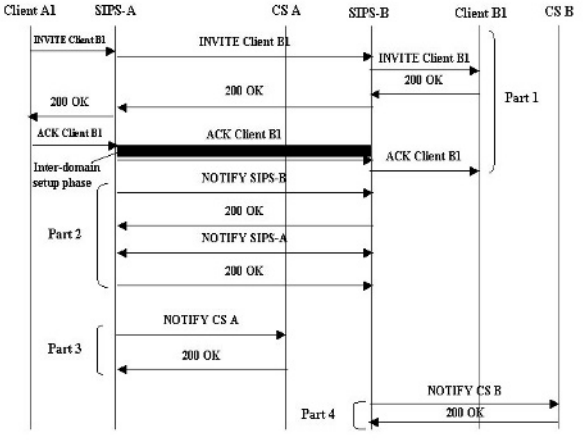

Fig. 6. Flow of SIP messages when Client A1 invites Client B1

5 Conclusion

In this paper, we have presented a new paradigm for VoIP conferencing. The fully distributed nature of SIPS and Conference servers make the scheme highly scalable. Multicast is not a mandatory requirement for our solution. However, it can be used with advantage, if available. The maximum number of hops for audio packets is limited to two (between two CSs), reducing the delay. The audio packets are transmitted using an RTP stack. The traffic is further reduced by VAD techniques [11], as an option. This paradigm, along with LN, supports an interactive conference without explicit floor control and allows impromptu speech while compensatory floor control co-exists. Further, this paradigm supports conferencing requirements listed in [8] and in section 2 of this paper. Moreover, we can support other facilities such as chat, PBX, and Voice Mail Service on the same set up. These facilities are based on a peer-to-peer communication mode. The main contribution of this work is a greatly improved

quality of conference due to the impromptu speech permitted without the use of 'gagging' floor controls. Further reduction of traffic is possible by using the characteristics LN. At present we are investigating the effective use of SCCP [1] (and CCCP) in our implementations.

References

[1] C. Bormann, J. Ott, "Simple Conference Control Protocol", Internet Draft, Dec. 1996.

[2] E. Doerry, "An Empirical Comparison of Copresent and Technologically-mediated Interaction based on Communicative Breakdown", Phd Thesis, Graduate School of the University of Oregon, 1995.

[3] H. P. Dommel and J.J. Garcia-Luna-Aceves, "Floor Control for Multimedia Conferencing and Collaboration", *J. Multimedia Systems*, Vol. 5, No. 1, January 1997, pp. 23–38.

[4] A. J. González, "A Distributed Audio Conferencing System" *MS Project Department Of Computer Science Old Dominion University*, Norfolk, VA 23529, July 28, 1997.

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

[6] M. Handley, J. Crowcroft et al., "Very large conferences on the Internet: the Internet multimedia conferencing architecture", *Journal of Computer Networks*, vol. 31, No. 3, Feb 1999, pp. 191–204.

[7] ITU-T Rec. H.323, "Packet based Multimedia Communications Systems", vol. 2, 1998.

[8] P. Koskelainen, H. Schulzrinne and X. Wu, "A SIP-based Conference Control Framework", *NOSSDAV'02*, May 2002, pp. 53–61.

[9] R Venkatesha Prasad et al., "Control Protocol for VoIP Audio Conferencing Support", *International Conference on Advanced Communication Technology*, Mu-Ju, South Korea, Feb 2001, pp. 419–424.

[10] R Venkatesha Prasad et al., "Automatic Addition and Deletion of Clients in VoIP Conferencing", *6th IEEE Symposium on Computers and Communications*, July 2001, Hammamet, Tunisia, pp. 386–390.

[11] R Venkatesha Prasad, H S Jamadagni, Abhijeet, et al "Comparison of Voice Activity Detection Algorithms" 7*th IEEE Symposium on Computers and Communications.* July 2002, Sicily, Italy, pp. 530–535.

[12] R. Venkatesha Prasad, Richard Hurni, H S Jamadagni, "A Scalable Distributed VoIP Conferencing using SIP", *To appear in Proc. of the 8th IEEE Symposium on Computers and Communications*, Antalya, Turkey, June 2003.

[13] R Venkatesha Prasad, H S Jamadagni and H N Shankar, "On Problem of Specifying Number of Floors in a Voice Only Conference", *To appear in Proc. of IEEE ITRE 2003*, Newark, NJ, Aug. 2003.

[14] A. B. Roach, " Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, IETF, June 2002.

[15] S. Ramanathan, P. Venkat Rangan, Harrick M. Vin, "Designing Communication Architectures for Interorganizational Multimedia Collaboration", *Journal of Organizational Computing,* Vol. 2, Nos. 3&4, 1992, pp.277–302.

[16] M. Radenkovic et al, "Scaleable and Adaptable Audio Service for Supporting Collaborative Work and Entertainment over the Internet", *SSGRR 2002*, L'Aquila, Jan. 2002.

[17] J. Rosenberg, H. Schulzrinne et al., "SIP", RFC 3261, IETF, June 2002.

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

[19] E. M. Schooler, "A Distributed Architecture for Multimedia Conference Control", Technical Report ISI/RR-91-289,USC/ISI, Marina del Rey, CA, Nov. 1991.

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

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