Deployment Issues of a VoIP Conferencing System in a Virtual Conferencing Environment

R. Venkatesha Prasad[i] Richard Hurni[ii] H.S. Jamadagni[iii] H.N. Shankar[iv]

[i, iii]Centre for Electronics Design and Technology
Indian Institute of Science, Bangalore, India
Telephone: +91 80 360 0810

[i, iii]{vprasad, hsjam}@cedt.iisc.ernet.in [ii]hurni@ieee.org [iv]hn_shankar@yahoo.com

ABSTRACT

Real-time services have been supported by and large on circuit-switched networks. Recent trends favour services ported on packet-switched networks. For audio conferencing, we need to consider many issues – scalability, quality of the conference application, floor control and load on the clients/servers – to name a few. In this paper, we describe an audio service framework designed to provide a *Virtual Conferencing Environment* (VCE). The system is designed to accommodate a large number of end users speaking at the same time and spread across the Internet. The framework is based on Conference Servers [14], which facilitate the audio handling, while we exploit the SIP capabilities for signaling purposes. Client selection is based on a recent quantifier called "Loudness Number" that helps mimic a physical face-to-face conference. We deal with deployment issues of the proposed solution both in terms of scalability and interactivity, while explaining the techniques we use to reduce the traffic. We have implemented a Conference Server (CS) application on a campus-wide network at our Institute.

Categories and Subjects Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *Client / Server, distributed applications.*

General Terms

Algorithms, Performance, Design, Theory.

Keywords

VCE, VoIP, Real-Time Audio, Simultaneous Speakers, SIP, Conference Server.

[ii] Swiss Federal Institute of Technology, Lausanne. Former visitor at CEDT.

[iv] PESIT and NIAS, Bangalore, India.

VRST'03, October 1-3, 2003, Osaka JAPAN.

1. INTRODUCTION

Today's Internet uses the IP protocol suite that was primarily designed for the transport of data and provides best effort data delivery. Delay-constraints and characteristics separate traditional data on the one hand from voice & video applications on the other. Hence, as progressively time-sensitive voice and video applications are deployed on the Internet, the inadequacy of the Internet is exposed. Further, we seek to port telephone services on the Internet. Among them, *virtual* conference (teleconference) facility is at the cutting edge. Audio and video conferencing on Internet are popular [25] for the several advantages they inhere [3,6]. Clearly, the bandwidth required for a teleconference over the Internet increases rapidly with the number of participants; reducing bandwidth without compromising audio quality is a challenge in Internet Telephony. Additional critical issues are: (a) packet delay, (b) echo, (c) mixing of audio from selected clients, (d) automatic selection of clients to participate in the conference, (e) playout of mixed audio for every client, (f) handling clients not capable of mixing audio streams (such clients are known as “dumb clients”), and (g) deciding the number of simultaneously active clients in the conference without compromising voice quality.

While all the above requirements are from the technology point of view, the user's perspective and interactions are also essential factors. There is plenty of discussion amongst HCI and CSCW community on the use of Ethnomethodology for design of CSCW applications. The basic approach is to provide larger bandwidth, more facilities and more advanced control mechanisms, looking forward to better quality of interaction. This approach ignores the functional utility of the environment that is used for collaboration. Eckehard Doerry [4] criticizes this approach by saying "it is keeping *form* before *function*". Thus, the need is to take an approach that considers both aspects – the *technical* and the *functional*. Regarding the functional aspect, we refer to [15] where it has been dealt with in some detail.

In this work, we do not discuss video conferencing; its inclusion does not significantly benefit conference quality [4]. Our focus is on virtual audio environments.

We first outline the challenges encountered in virtual audio conferences. Then we look into the motivations followed by relevant literature. In Section 5, we explain the architecture of our system. Section 6 comprises description of the various algorithms used in our setup. We address deployment issues. A discussion on

performance follows. We conclude taking alongside some implementation issues.

2. CHALLENGES IN VoIP CONFERENCING

Many challenges arise in building a VoIP application. The following are of particular concern in the process:

- *Ease of use*: Conferencing must be simple; users need no domain expertise. Management (addition/removal) of clients and servers must be uncomplicated. Application development should not presuppose specific characteristics of the underlying system or of network layers. Ease of use may include leveraging readily available, technically feasible and economically viable technologies.
- *Scalability*: Conferencing must seem uninterrupted under heavy loads, i.e., when many additional users are added on. Traffic on WAN should not grow appreciably with the total number of clients; else, this has lead to congestion. So a means to regulate traffic to a minimum is needed for this kind of real-time applications.
- *Interactivity*: In Virtual Conferencing Environments (VCEs), we intend a face-to-face-like conferencing application that mimics a "real" conference, where more vocal participants invite attention. Turn-taking in floor occupation by participants must be adapted gracefully to give a feel of natural transition.
- *Standardization*: The solution must conform to established standards so as to gain interoperability and peer acceptance.

The above requirements are placed in the perspective of observations made in earlier works (vide Sections 3 and 4) and will steer the VCE design.

3. THE MOTIVATION

Ramanathan and Rangan [20] have studied in detail the architectural configurations comparing many conferencing architecture schemes taking into consideration the network delay and computation requirements for mixing. Functional division and object-oriented architecture design that aid in implementation is presented in [1]. An overview of many issues involved in supporting a large conference is dealt in [8]. H. P. Dommel [5] and many others highlight floor control as another pivotal aspect to be taken into account in designing a conferencing tool. Tightly coupled conference control protocols in Internet belong to the ITU-T H.323 family [9]; however, they are mainly for small conferences. The latest IETF draft by Rosenberg and Schulzrinne [23] discusses conferencing models with SIP [22] in the background. Aspects of implementation for centralized SIP conferencing are reported in [26]. A new approach called partial mixing by Radenkovic [18] allows for mixed and non-mixed streams to coexist. In all the above proposals, while there are some very useful suggestions, they share one or more of the following limitations:

- In an audio conference, streams from all the clients need not be mixed. Actually, mixing many arbitrary streams [24] from clients degrades the quality of the conference due to the reduction in the volume (spatial aspect of speech). The number of streams mixed varies dynamically depending on the number of active participants. This would lead to fluctuations in the volume of every individual participant causing severe degradation in quality. Customized mixing of streams is not possible when many clients are active. There is a threshold on the number of simultaneous speakers above which increasing the number of speakers becomes counterproductive to conference quality. Fixing the maximum *number* of simultaneous speakers is dealt in a recent work [15] using Ethnomethodology, and is conjectured to be *three*. Thus it is advisable to honour that constraint.
- There cannot be many intermediate mixers (similarly, Conference Servers as in [10]) in stages as in [20] because it brings in inordinate delay by increasing the number of hops and is not scalable with interactivity in focus.
- Floor Control for an audio conference (even video conference) with explicit turn-taking instructions to participants renders the conference essentially a one-speaker-at-a-time affair, not a live and free-to-interrupt one. This way, the conference becomes markedly artificial and its quality degrades. Schulzrinne et al. [24], assume only one participant is speaking at a time. In this case, if applications are implemented with some control [5], the service becomes 'gagging' for the users.
- Partial mixing [18] has a similar problem as that of mixing when more streams are mixed. Moreover, in [18], to allow impromptu speech, mixing is not done when the network can afford high bandwidth requirements for sending/receiving all the streams, but it is unnecessary [15].
- For large conferences [23, 10] a centralized conference cannot scale up. With multicasting, clients will have to parse many streams and traffic on a client's network increases unnecessarily.

Evidently, different particular issues, all of which are a subset of requirements (defined in [14] and [16]) for a VoIP conferencing support, are tackled. Thus there is a need to address *conferencing* as a whole with all its requirements considered concurrently. Towards this goal, the VoIP conferencing system we propose is intended to be scalable and interactive. We make use of the "Loudness Number" for implementing floor control. This permits a participant to freely get into the speaking mode to interrupt the current speaker as in a natural face-to-face meeting. An upper limit on the number of floors (i.e., the number of speakers allowed to speak at the same time) is fixed using a conjecture proposed in [15].

The work presented here is in continuation of our studies into conferencing based on the Session Initiation Protocol in [14] and [16]. SIP, defined in [22] is now the most popular standard for VoIP deployment and has been chosen for its strength, ease of use, extensibility and compatibility. This is the reason it will be in the background of all controlling messages that will implicitly arise between the entities in our architecture. The actual messages are described in [16] and, as such, we do not present a complete description of them here.

4. RELATED WORK

The SIP standard defined in RFC 3261 [22] and in later extensions such as [21] does not offer conference control services such as floor control or voting and does not prescribe how a

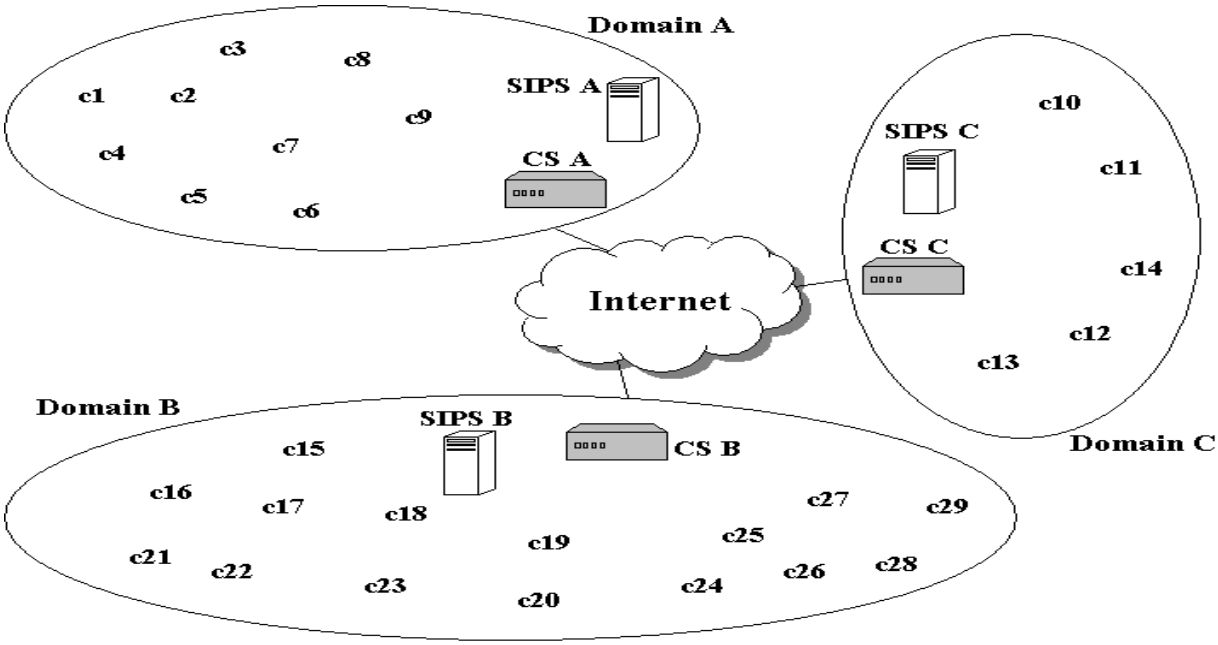

Fig. 1. Conference example – 3 domains containing the necessary entities so that the conference can take place.

conference is to be managed. However SIP can be used to initiate a session that uses some other conference control protocol.

The core SIP specification supports many models for conferencing [26, 23]. In the server-based models, a server mixes media streams, whereas in a server-less conference, mixing is done at the end systems. SDP [7] can be used to define media capabilities and provide other information about the conference. We shall now consider a few conference models in SIP that have been proposed recently [23].

First, let us look into server-less models. In *End-System Mixing*, only one client (SIP UA) handles the signaling and media mixing for all the others, which is clearly not scalable and causes problems when that particular client leaves the conference. In the *Users Joining* model, a tree grows, as each invited party constitutes a new branch in the distribution path. This leads to an increasing number of hops for the remote leaves and is not scalable. Another option would be to use multicast for conferencing but multicast is not enabled over Internet and only possible on a LAN presently. Among server-based models, in a *Dial-In Conference*, UAs connect to a central server that handles all the mixing. This model is not scalable as it is limited by the processing power of the server and bandwidth of the network. *Ad-hoc Centralized Conferences* and *Dial-Out Conference Servers* have similar mechanisms and problems.

Hybrid models involving centralized signaling and distributed media, with the latter using unicast or multicast, raise scalability problems as before. However an advantage is that the conference control can be a third party solution.

Distributed Partial Mixing, presented in [18], proposes that in case of bandwidth limitation, some streams are mixed and some are not, leaving interactivity intact. Loss of spatialism when they mix and the bandwidth increase when they do not are open problems. A related study [19] by the same author proposes conferencing architecture for Collaborative Virtual Environments (CVEs) but does not provide the scalability angle without the availability of multicasting. With the limitations of proposed conferencing systems in mind, we will now detail our proposal.

5. SYSTEM ARCHITECTURE

This section is dedicated to the description of the proposed system architecture. However, as this paper constitutes the continuation of our work started in [14] and furthered in [16], we will not present here all the details about the proposed entities and invite the readers to consult the papers mentioned above for a full and thorough description.

First, we do not restrict our conferencing system to work on small conferences only, but rather on large audio VCEs that have hundreds (or even thousands) of users across a Wide Area Network (WAN) such as the Internet. This view stems from an appraisal that VCEs will gain in importance in the future, as interactive audio conferences will be more popular because of the spread of the media culture around the world.

Two issues must be taken care of when building a VoIP conferencing system: (i) the front-end, consisting of the application program running on the end-users' computers and (ii) the back-end that provides other application programs that facilitate conferencing and conference.

The participating users are grouped into several *domains*. These domains are Local Area Networks (LANs), such as corporate or educational networks. This distributed assumption asks for distributed controlling and media handling solutions, as centralized systems would not scale for such very large conferences (vide Section 4).

More explicitly, in each domain, we can identify several relevant logical components of a conferencing facility (Fig. 1):

- An arbitrary number of end users (clients) that can take part in at most one audio conference at a time. Every user is

included in one and only one domain at a given instant, but can move from domain to domain (nomadism). In our conferencing environment, these clients are regular SIP User Agents (SIP UAs), as defined in [22] so to gain in interoperability with other existing SIP-compatible systems. These clients are thus not aware of the complex setting that supports the conference and this is highlighted below.

- One SIP Server (SIPS) per domain, taking care of all the signaling aspects of the conference (clients joining, leaving, etc.) [16]. In particular, it is considered as a physical implementation encompassing different logical roles, namely a SIP Proxy Server, a SIP Registrar Server, a SIP Redirect Server and a SIP B2BUA (Back-to-Back User Agent) [22]. This physical implementation enables the handling of incoming/outgoing SIP messages by one or another logical entity according to the needs. SIPS is entrusted with maintaining total service and has many advantages such as (a) it works as a centralized entity that can keep track of the activities of the UAs in a conference; (b) it can do all the switching for providing PBX features; (c) it can locate the UAs and invite them for a conference; (d) it can do the billing as well. SIPSs in different domains communicate with each other using SIP messages as described in [16]. If the load on a particular SIPS is too heavy, it can create another SIPS in the same domain so that the load will be shared.
- One Master Conference Server (M-CS) (simply a Conference Server (CS)) for each conference is created by the local SIPS when a conference starts. This server will be used for handling media packets for the clients of the domain. Its mechanism will be described in the next section. The M-CS will be able to create a hierarchy of CSs inside a domain by adding one or more Slave CSs (S-CSs) to accommodate all the active clients and prevent its own flooding at the same time. We will see this mechanism in some detail in the sequel.

The entities described here are exhaustive and conform to the SIP philosophy. Thus, the use of SIP makes this architecture more useful and interoperable with any other SIP clients or servers.

6. ALGORITHMIC ISSUES

6.1 Selecting the Streams

Similar to SipConf in [27], a Conference Server (CS) [17] has the function of supporting the conference; it is responsible for handling audio streams using RTP. It can also double to convert audio stream formats for a given client if necessary and can work as *Translators/Mixers* of RTP specification behind firewalls.

We have based the design of our CS on H.323 Multipoint Processor (MP) [9]. In short, the MP receives audio streams from the endpoints involved in a centralized or hybrid multipoint conference, processes them and returns them to the endpoints. An MP that processes audio prepares N_{Max} audio outputs from M input streams after selection, mixing, or both. Audio mixing requires decoding the input audio to linear signals (PCM or analog), performing a linear combination of the signals and re-encoding the result in an appropriate audio format. The MP may eliminate or attenuate some of the input signals in order to reduce noise and unwanted components.

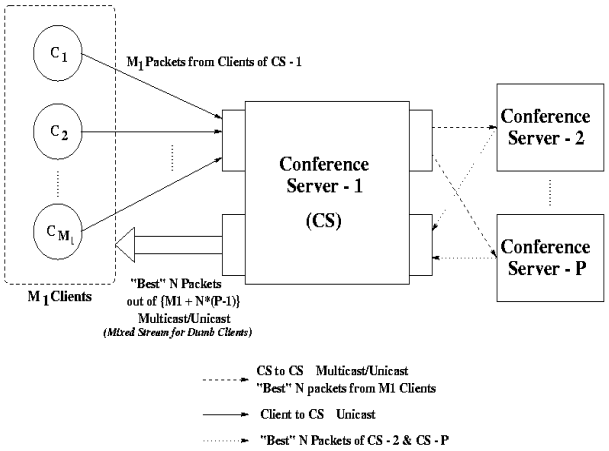

Fig. 2. Schematic diagram of a CS

The limitation of H.323 is that it does not address the scalability of a conference. The architecture proposes a cascaded or daisy chain topology [10], which can be shown that it cannot scale up for a large conference.

A CS serves many clients in the same conference. Thus it handles only one conference at a time. Multiple CSs may coexist in a domain, as when there are several conferences under way. Signaling-related messages of CSs are dealt in [11].

The working of a CS is illustrated on Fig. 2: For each mixing interval, CS 1 chooses the "best" $\boldsymbol{N_{Max}}$ audio packets out of the M_1 (using a criterion termed "Loudness Number", described in the next subsection). It may possibly receive and sends these to CSs 2 to P. The set of packets sent is denoted as "*ToOtherCSs*". In the same mixing interval, it also receives the best $\boldsymbol{N_{Max}}$ audio packets (out of possibly M_2) from CS 2, similarly the best $\boldsymbol{N_{Max}}$ (out of possibly M_P) from CS P. For simplicity, we ignore propagation delay between CSs which indeed can be taken into account; it is beyond the scope of this presentation. The set of packets received is denoted as "*FromOtherCSs*". Finally, it selects the best $\boldsymbol{N_{Max}}$ packets from the set {*ToOtherCSs union FromOtherCSs*} and passes these packets to its own group.

It can be seen that the set {*ToOtherCSs union FromOtherCSs*} is the same at all CSs. This ensures that any client in the conference finally receives the same set of packets for mixing. Hence all clients obtain a common view of the conference.

Similarly, for each time slot (packet time), a subset, F of all clients is selected (using the same criterion) from the pool of packets from all other CSs plus the N_{Max} clients selected locally. Their packets are mixed and played out at the clients. According to [15], the cardinality of F, $|F|$ is N_{Max} and is fixed at three.

In our conferencing setup, selection is by the Master Conference Server (M-CS), which comes into the picture exclusively for media handling. Note that 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).

There are cases wherein the processing capacity of an M-CS is exceeded as it may have too many packets – from local domains and from remote domains – to process. In that case, the M-CS will create one or many S-CS (Fig. 6) and transfer its own clients as well as the new clients to them. In this configuration, the

algorithm outlined above will be slightly modified, as the audio packets will go from clients to their dedicated S-CS that will select N_{Max} packets to send to the local M-CS, which will then select N_{Max} packets from all its S-CSs in the domain before sending them to the remote domains. The incoming packets from other domains will be received by the M-CS, select N_{Max} of them and send them directly to the domain clients, bypassing the S-CSs. This change implies that at most three intermediate entities exist for each audio packet, instead of two in the conventional setup. As the extra hop happens inside the LAN supposed to have a high-speed connectivity, we consider that it should not prevent us from using this hierarchy of CSs when there's a need to do so.

6.2 Loudness Number (LN)

A basic question to be answered by the CS is the following. In a mixing interval, how should it choose N_{Max} packets out of the M it might possibly receive? One way is to rank the M packets received according to their energies, and choose the top N_{Max}. However, 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. The metric should have the following characteristics [12]:

- A speaker (floor occupant) should not be cut off by a spike in the packet energy of another speaker. This implies that a speaker's speech history should be given some weight. This is often referred to as "Persistence" or "Hangover".
- A participant who wants to interrupt a speaker will have to (i) speak loudly and (ii) keep trying for a little while. In a face-to-face conference, body language often indicates the intent to interrupt. But in a blind conference under discussion, a participant's intention to interrupt can be conveyed effectively through LN.

A floor control mechanism empowered to cut off a speaker forcefully must be ensured.

These requirements are met by Loudness Number [12], which changes smoothly with time so that the selection (addition and deletion) of clients for conference is graceful.

LN (=λ) is a function of the amplitude of the current audio stream plus the activity and amplitude over a specific window in the past.

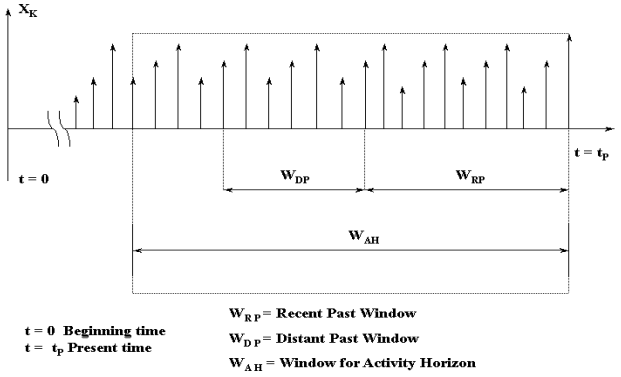

Fig. 3. The different windows used for LN computation

The Loudness Number is updated on a packet-by-packet basis. The basic parameter used here is packet amplitude, which is calculated as root mean square (rms) of the energies in audio samples of a packet, and denoted by X_K. Three windows are defined as shown in Fig. 3.

The present amplitude level of the speaker is found by calculating the moving average of packet amplitude (X_K) within a window called "*Recent Past Window*" starting from the present instant to some past time. The past activity of the speaker is found by calculating the moving average of the packet amplitude (X_K) within a window called "*Distant Past Window*", which starts at the point where the "Recent Past" window ends and stretches back in the past for a pre-defined interval. The activity of the speaker in the past is found with a window called "*Activity Horizon*", which spans the recent past window as well as the distant past window and beyond if necessary. Though the contribution of the activity horizon looks similar to the contribution of the recent past and distant past windows, past activity is computed from activity horizon window in a differently.

Define the quantities during these three intervals as L_1, L_2 and L_3. L_1 quantifies the Recent Past speech activity, L_2 the Distant Past speech activity and L_3 gives a number corresponding to the speech activity in the Activity Horizon window quantifying how active the speaker was in the past few intervals. L_3 yields a quantity that is proportional to the fraction of packets having energies above a pre-defined threshold (Eq. 3). The threshold is invariant across clients.

$$L_1 = \frac{1}{W_{RP}} \sum_{K=t_P}^{t_P - W_{RP}+1} X_K \tag{1}$$

$$L_2 = \frac{1}{W_{DP}} \sum_{K=t_P - W_{RP}}^{t_P - W_{RP} - W_{DP}+1} X_K \tag{2}$$

$$L_3 = \frac{1}{W_{AH}} \sum_{K=t_P}^{t_P - W_{AH}+1} \theta * I_{\{X_K \geq \theta\}} \tag{3}$$

Where $I_{\{X_K \geq \theta\}} = 1 \; if \;\; X_K \geq \theta$

$= 0, otherwise$

The threshold θ is a constant. θ is set at 10-20 percent of the amplitude of the voice samples of a packet in our implementation here. Loudness Number λ for the present time instant (or the present packet) is calculated as,

$$\lambda = \alpha_1 * L_1 + \alpha_2 * L_2 + \alpha_3 * L_3 \tag{4}$$

Here α_1, α_2 and α_3 are chosen such that:

$$0 < \alpha_1, \alpha_2 < 1, \; 0 < \alpha_1 + \alpha_2 < 1 \text{ and } \alpha_3 = 1 - (\alpha_1 + \alpha_2)$$

Here, α_1 is the weight given to the recent past speech, α_2 is the weight given to distant past speech and α_3 is the weight given to speech activity in the activity horizon window considered.

6.3 Safety, Liveness and Fairness

The λ parameter has some memory depending on the spread of the windows. After one conferee becomes silent, another can take the floor. Also, as there is more than one channel, interruption is enabled. A loud conferee is more likely to be heard because of elevated λ. This ensures fairness to all conferees. After all, even in a face-to-face conference, a more vocal speaker grabs special attention. All these desirable characteristics are embedded into the LN. A comprehensive discussion on selection of the various parameters and the dynamics of LN are beyond the scope of this paper.

6.4 Selection Algorithm using the LN

Following the developments in subsections 6.1 and 6.2, we present the simple algorithm that runs at each Master-Conference Server (Algorithm. 1). This algorithm is based on the discussions in section 6.1. The globally unique set *F* is found using this procedure.

```
Repeat for each time slot at each M-CS
{
   1. Get all the packets from the Clients
      that belong to it.
   2. Find at most N_Max Clients that have
      maximum λ out of M Clients in its domain.
   3. Store a copy of packets from those N_Max
      Clients in database DB_1.
   4. Send these N_Max packets to other M-CSs (on
      Unicast or Multicast, depending on the
      configuration).
   5. Similarly, receive packets from all
      other M-CSs and store them in database
      DB_2.
   6. Now compare the packets in DB_1 and DB_2 on
      the basis of λ and select a maximum of
      N_Max amongst them (to form set F) that
      should be played out at each Client.
   7. Send the N_Max packets in set F to the
      Clients in its domain.
   8. Mix these N_Max audio packets in set F after
      linearising and send it to dumb Clients
      in the domain.
}
```

Algorithm 1. Selection algorithm

The mechanism proposed here is also depicted on Fig. 6, where a single conference takes place between three domains. The shaded clients are the ones selected in their local domains; their audio streams will be sent to other CSs.

7. DEPLOYMENT ISSUES

We now analyze deployment issues associated with conference management. How are domains to be organized to maximize the number of participants able to join? To address this, we define some useful parameters.

- Let *d* be the number of different domains in which there are active clients in a given conference.
- Let M_i be the number of active clients present in domain *i* ($1 \le i \le d$) in a given conference. The total number of active clients in the conference is thus $M = \sum_{i=1}^{d} M_i$.
- Let *C* be the maximum number of audio streams a Conference Server can handle in a packet time, also called capacity. *C* is set according to the processing power of the weakest CS in the conference but as it cannot be assumed that we know it a-priori, it can be set according to some minimum system requirement a machine must meet in order to take part in a conference.
- Let N_{Max} be the number of output streams a CS has to send to other CSs in remote domains (see section 6.1). We will set $N_{Max} = 3$ (=|*F*|), according to [15].

The optimization problem is now to find the value of *d* that maximizes the total number of clients M_i served by one CS in a domain with capacity *C*. We first dispose the case where the capacity is not exceeded (the existing CS is not overloaded), and then proceed to the case where there exists a need to create more CSs when a single CS is overloaded.

We assume that clients are equally distributed amongst the domains, as we may not have information to assume an a-priori distribution of the clients. We can specify no more than an upper bound on the number of clients acceptable, given the number of active domains *d*.

7.1 Conferencing with only One Level of CSs

In this subsection, we consider that we have only one CS, i.e., a unique M-CS in each domain. Thus it cannot be overloaded. We consider that the system works as outlined in section 6.1: The Clients send their audio packets to their local CS, which selects N_{Max} streams, before sending them to other CSs. In parallel, it also receives N_{Max} streams for every other CSs before taking a decision on which N_{Max} streams will be selected, sent and played out at each individual clients.

For system stability, any CS in the conference should be able to handle its local clients in addition to the audio packets from other domains. Clearly then, the following inequality must hold for every domain:

$$C \ge \frac{M}{d} + N_{Max} \cdot (d-1) \tag{5}$$

The limiting case of (5) (taking the equality) takes the form

$$M = (C + N_{Max}) \cdot d - N_{Max} \cdot d^2 \tag{6}$$

To optimize *d* with respect to *M*, we set

$$\frac{\partial M}{\partial d} = 2 \cdot N_{Max} \cdot d - (C + N_{Max}) = 0 \tag{7}$$

yielding

$$d = \left[\frac{C + N_{Max}}{2 \cdot N_{Max}} \right] \tag{8}$$

($[*]$ = Rounding to nearest integer) and hence, *M* from (6).

C	***d***	***M***
50	9	234
100	17	884
150	26	1950
200	34	3434
250	42	5334
300	51	7650
350	59	10384
400	67	13534
450	76	17100
500	84	21084

Table 1. Values of *d* and *M* computed for some values of *C* with N_{Max} = 3.

In Table 1, we give the values of *d* and *M* that were computed using (8) and (6) with N_{Max} = 3. We see that the values of *d* and *M*, being dependent of *C*, are therefore based on the weakest CS. We see that there is a trade-off between *M* and *d*. We could admit

more domains in the conference, but at the expense of restricting the total number of clients *M* in the conference.

While implementing and testing the Conference Servers on a Pentium III 1.4 GHz running Windows NT, we were able to set *C*=300. But with the advent of faster computers (> 3 GHz), one can easily set *C* to higher values and determine *d* and *M* accordingly.

Fig. 4 shows a contour plot and Fig. 5, a 3D-mesh showing optimized solutions for CSs of different capacities. These lead us to maximize the number of domains, and hence, to maximize the total number of clients based on the capacity of various CSs. In Fig. 4, the individual curves represent the total number of clients targeted, and we select a lower value of *d*, for capacity *C*, for targeted *M* to reduce traffic on WAN. Fig. 5 represents a different perspective of the same data in 3D.

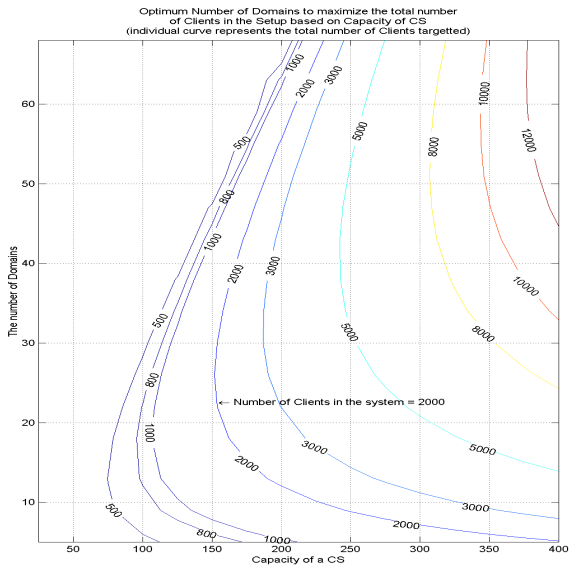

Fig. 4. Contour Plot of Capacity versus Optimum number of domains for various conference sizes

7.2 Conferencing with Two Levels of CSs

Now considering the case where the number of clients in a particular domain is too large, i.e., $M_i \geq \frac{M}{d}$ (9)

one has to avoid the denial of service for new clients due to overloading of Conference Server. This problem can be solved by introducing a second level of CSs inside the given domain, as in Fig. 6. The existing M-CS creates a Slave CS (S-CS) that can handle up to *C* end-users and to which it transfers all its active clients. Here, the system works differently as outlined in section 6.1: The Clients send their audio packets to their local S-CS, which selects N_{Max} streams, before sending them to a local M-CS, which will proceed in the same way, before sending N_{Max} streams to the other domains. Each newly created S-CS must run on a separate machine. The M-CS has to create more S-CSs if the number of active clients exceeds *C* in the course of the conference after the transfer.

With this mechanism, the M-CS will be able to create utmost

$$U = \left\lfloor \frac{C - N_{Max} \cdot (d-1)}{N_{Max}} \right\rfloor \text{S-CSs,} \quad (10)$$

as it must handle 3 (= N_{Max}) packets for each local S-CSs and 3 (= N_{Max}) packets from each other remote domains. We can then calculate the maximum theoretical number of active clients $M_i = U \cdot C$ in each domain as well as *M*, for the whole conference as $M = d \cdot U \cdot C$.

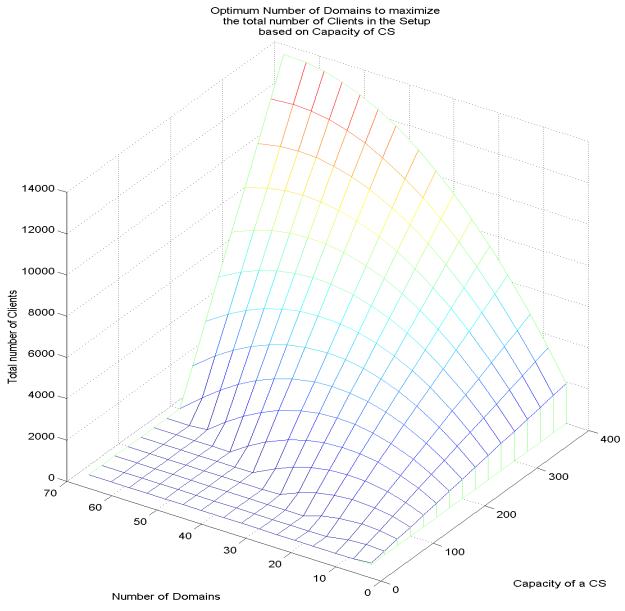

Fig. 5. 3D Plot of Capacity versus Optimum number of domains for various conference sizes

Of course, one could further create a third level in the hierarchy, giving the possibility of accommodating even more clients. This may be unnecessary as the number of possible clients is large enough with two levels.

8. PERFORMANCE DISCUSSION

We now analyze the performance of the algorithm presented in subsection 6.3, i.e., the one taking care of the exchange of audio packets between the different domains. Note that the packets that are transiting within the LAN take advantage of the higher capacity (generally coupled with multicast capabilities) and therefore do not require a performance analysis.

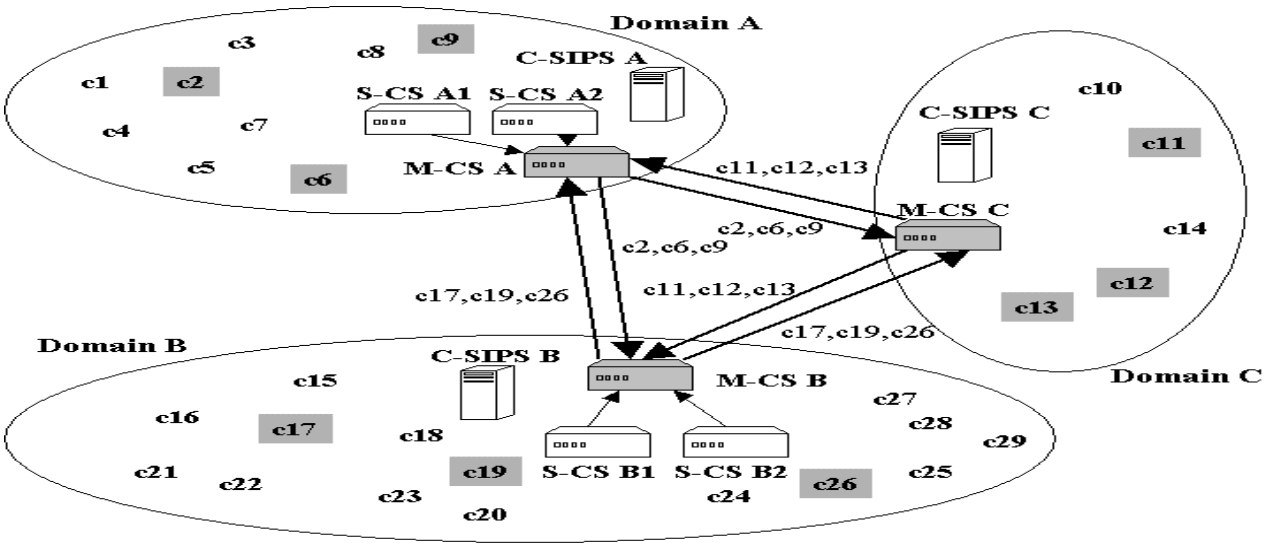

Fig. 6. Example of a 2-level hierarchy of Conference Servers; the shaded Clients are the one selected by the M-CS and will be sent to other domains' CSs.

Thus we have to look only at the RTP packets over the WAN, i.e., between participating M-CSs. As each M-CS from a domain will be sending only N_{Max} out of $\frac{M}{d}$ packets to the other CSs ($\frac{M}{d} >> N_{Max}$), the bandwidth used by the application over a WAN is upper-bounded by the following expression.

The total number of audio packets transiting over the WAN for each time slot is $\sum_{i=1}^{d}\left(\sum_{j=1, j\neq i}^{d} N_{Max}\right)$ which is quadratic in the number of domains (i.e., $O(d^2)$).

However, it is independent of the total number of active clients. This would not have been the case had all packets been sent over the network in each time slot.

The saving is tremendous. Yet, one may contend that sending three packets to and from all domains is a waste of resources, as most of these streams will not be selected. If just one client is active, selecting a subset of clients in that domain is unnecessary. Pessimistic and optimistic algorithms as presented in the sequel aim at reducing the traffic further by harnessing the slow varying nature of the LN.

8.1 Pessimistic algorithm

Consider a scenario wherein the lowest LN (called LN_t) of the three globally selected streams (set *F* of Section 6.1) exceeds the LN of the most dominant stream of a domain. Evidently, the chances that the next two dominant streams of that domain being selected to *F* in the next packet period are less. Here, we send this most dominant stream and withhold the other two. There may be an error in unique selection across all domains for one packet period only. As LN varies slowly, the error would get automatically rectified in a subsequent packet period (slot). In this algorithm, there is at least one stream in each period. The net network traffic in a packet period in the best case is $d \cdot (d-1)$, i.e., $O(d^2)$ using unicast, instead of $d \cdot (d-1) \cdot N_{Max}$. Considerable valuable bandwidth can be saved using this heuristic. The resulting traffic complexity reduces from $O(d^2)$ to $O(d)$ in multicast-enabled networks.

```
Initialize LN_t = 0 at an M-CS/S-CS
A.  In the first time slot (packet time), each CS
    sends the top N_Max streams (based on their LN)
    to all other CSs.
At each M-CS/S-CS and for each packet time:
B.  Find the value of lowest LN of the N_Max
    globally selected streams (set F) from the
    previous time slot. Set LN_t with this value.
C.  At each CS domain, select the N_Max local
    streams that have maximum value of LN
    (ToOtherCSs set).
D.  Select streams that have LN > LN_t.
    IF there are >= N_Max streams with LN > LN_t
       then send top N_Max to other CSs.
    ELSE IF there are (N_Max-1) streams with LN>LN_t
       then send top (N_Max-1) plus the one lower
       than LN_t (i.e., top N_Max) to other CSs.
    ELSE IF there are (N_Max-2) streams with LN>LN_t
       then send top (N_Max-2) plus the one lower
       than LN_t (i.e. top (N_Max -1)) to other CSs.
    ......
    ELSE IF there are NO streams with LN> LN_t
       then send top 1 stream to other CSs.
E.  Packets sent in step D form DB_1. Packets
    received from other CSs form DB_2.
F.  For this time slot, find global N_Max streams
    based on LN from DB_1 U DB_2 (set F)
G.  Send set F to the clients in its domain.
    Update LN_t for the next period.
```

Algorithm 2. Pessimistic algorithm to reduce the number of packets sent over the Internet.

In this algorithm the saving in traffic is at the cost of relaxing the condition of formation of globally unique set *F*. However, the discrepancies in selected streams at different domains remain for a short period of time depending on the transportation delay between any two domains. Even for a total delay of 400ms, for only 10 packet time slots the uniqueness is lost. This duration in a real-time interactive conversation is non-perceivable by the listener. In the case that there is a joke and every one laughs, then there would be sudden rise in the number of packets and it would be upper bounded by $O(d^2)N_{Max}$ for a short period.

8.2 Optimistic Algorithm

The traffic can be reduced further. The scheme in the following algorithm (Algorithm. 3) is withholding all the streams that have less value of LN compared to the least of the three in the set *F*. We can find the correct and unique three streams after a few time slots depending on the transportation delay between the domains. As the packet period is of the order of 40ms, the error in the selection is unnoticeable. The number of streams on network in this case is always restricted to N_{Max} (=3). Even without Voice Activity Detection (VAD), there will be no more than three streams in the network in the best case, thus the total traffic is constant. A sudden burst of traffic, as described in 8.1, is a particular case. These advantages are due to exploitation of the characteristics of LN.

```
Initialize LNt = 0 at an M-CS/S-CS
A.  In the first time slot (packet time), each CS
    sends the top NMax streams (based on their LN)
    to all other CSs.
At each M-CS/S-CS and for each packet time:
B.  Find the value of lowest LN of the NMax
    globally selected streams (set F) from the
    previous time slot. Set LNt with this value.
C.  At each CS domain, select the NMax local
    streams that have maximum value of LN
    (ToOtherCSs set)
D.  Select streams that have LN > LNt
    IF there are >= NMax streams with LN > LNt
       then send top NMax to other CSs.
    ELSE IF there are (NMax-1) streams with LN>LNt
       then send top (NMax-1) and see E.
    ELSE IF there are (NMax-2) streams with LN>LNt
       then send top (NMax-2) and see E.
    ......
    ELSE IF there are NO streams with LN> LNt
       then don't send any stream.
E.  Exceptions:
    IF the stream that was in F in the last
    interval belongs to this CS then select and
    send that stream even if its LN is now < LNt.
    (Note this occurs only at that CS which had
    the stream that was the last of the three in
    the previous packet period.)
F.  Packets sent in step D and E form DB1. Packets
    received from other CSs form DB2.
G.  For this time slot, find global NMax streams
    based on LN from DB1 U DB2 (set F).
H.  Send set F to the clients in its domain.
    Update LNt for the next period.
```

Algorithm 3. Optimistic algorithm to reduce the number of packets sent over the Internet

Furthermore, when VAD is used [13], it would further reduce the traffic by sending the header part of the RTP packet only and not the whole packet, thus in order to keep updating the LN across. The traffic here in this case is $O(N_{Max})$ for multicast and $O(d)$ for unicast.

We see that the above algorithms save bandwidth and computation at each CS, and leads to a scalable architecture with multiple CSs mainly because clients are grouped in domains. The necessary bandwidth is not dependent on the total number of active clients. As the CS always chooses the best three 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.

8.3 Availability of Multicasting

In the architecture that has been proposed, no assumption was made about the availability of multicasting support from the network. The traffic will be further reduced if *multicasting* is available over WAN. It is simple to show that the order of traffic would tend to become $O(d)$ from $O(d^2)$. This is an approximation as saving in multicasting depends also on the topology. The analysis was done for the case wherein multicast is not available (a realistic assumption in today's Internet). The advantage of this set up is that we can use it even if multicasting is partially available. We can instruct CSs during the set-up phase to send unicast packets to those CSs that cannot receive multicast packets whereas CSs on multicast enabled routers can exchange packets on a multicast address. The data structures and conference objects inside a CS is given in [14].

Fig. 7. User Interface for setting the weight for N_{Max} audio streams (setting Self-bar to zero avoids echo).

8.4 Quality Improvement

The observed improvement in the perceived quality of the conference service is due to: (1) limiting the number of concurrent speakers to a low number such as three. Generally, in a conference if more than two participants speak the intelligibility is lost. The conversational analysis demonstrates that there would be a repair mechanism [15] in such a case. (2) Delay: The audio stream between any two clients will pass through at most two CSs thus reducing the end-to-end delay. For a large conference there might be three CSs however, one hop is within the domain

incurring negligible delay. (3) As the streams are mixed only at the clients, there can be a customized mix of the streams. The individual tuning of mixing with weights the spatialism is preserved. Fig. 7 shows the user interface for the same. The echo when self-stream is selected can be avoided by reducing the weight. Nonetheless, feedback helps in reassuring speaker that he/she is heard by all.

9. CONCLUSION

In this paper, we have presented a discussion on a voice-only virtual conferencing environment. We have argued that the distributed nature of deployment here makes it scalable. Interactivity is achieved by adapting a recent stream selection scheme based on Loudness Number. Additionally, we incorporate a result from a more recent work [15] where the sufficiency of three simultaneous speakers has been demonstrated. Thus, there is significantly effective utilization of bandwidth. A mixed stream is played out at each client; each client may choose to have a customized mix since mixing is done at the local terminal of each client. These render impromptu speech in a *virtual* teleconference over VoIP a reality, as in a real face-to-face conference. Compatibility is assured thanks to the use of SIP, the most sought-after signaling protocol. To ensure a satisfying performance, we do not demand the availability of multicast, but use it if and when available. The traffic in the WAN (Internet) is upper-bounded by the square of the number of domains, -- further reduced by using heuristic algorithms -- which is far below the total number of clients in the conference. This is due to the use of a Conference Server local to each domain. VAD techniques help further traffic reduction. Using SIP standard for signaling makes this solution highly interoperable.

We have implemented a CS application on a campus-wide network. We believe this new generation of virtual conferencing environments will gain more popularity in the future as their ease of deployment is assured thanks to readily available technologies and scalable frameworks.

10. REFERENCES

[1] L Aguilar et al., “Architecture for a Multimedia Teleconferencing System”, in *Proceedings of the ACM SIGCOMM*, Aug 1986, pp. 126-136.

[2] Carsten Bormann, Joerg Ott et al., “Simple Conference Control Protocol”, Internet Draft, Dec. 1996.

[3] M. Decina and V. Trecordi, "Voice over Internet Protocol and Human Assisted E-Commerce", *IEEE Comm. Magazine*, Sept. 1999, pp. 64-67.

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

[5] 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.

[6] Amitava Dutta-Roy, "Virtual Meetings with desktop conferencing", *IEEE Spectrum*, July 1998, pp. 47-56.

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

[8] 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.

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

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

[11] 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.

[12] 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.

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

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

[15] 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 *IEEE ITRE 2003*.

[16] R. Venkatesha Prasad, Richard Hurni, H S Jamadagni, "A Proposal for Distributed Conferencing on SIP using Conference Servers", To appear in the *Proc. of MMNS 2003*, Belfast, UK, September 2003.

[17] R. Venkatesha Prasad, H.S. Jamadagni, J. Kuri, R.S. Varchas, “A Distributed VoIP Conferencing Support Using Loudness Number”, Tech. Rep. TR-CEDT-TE-03-01

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

[19] M. Radenkovic, C. Greenhalgh, S. Benford, “Deployment Issues for Multi-User Audio Support in CVEs”, *ACM VRST 2002*, Nov. 2002, pp. 179-185.

[20] Srinivas Ramanathan, P. Venkata Rangan, Harrick M. Vin, “Designing Communication Architectures for Interorganizational Multimedia Collaboration”, *Journal of Organizational Computing*, 2 (3&4), pp.277-302, 1992.

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

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

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

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

[25] Lisa R. Silverman, "Coming of Age: Conferencing Solutions Cut Corporate Costs", White Paper, www.imcca.org/wpcomingofage.asp

[26] Kundan Singh, Gautam Nair and Henning Schulzrinne, "Centralized Conferencing using SIP", *Proceedings of the 2nd IP-Telephony Workshop (IPTel)*, April 2001.

[27] D. Thaler, M. Handley and D. Estrin, "The Internet Multicast Address Allocation Architecture", RFC 2908, IETF, Sept. 2000.