A Scalable Architecture for VoIP Conferencing

R Venkatesha Prasad*
Joy Kuri*

* CEDT, Indian Institute of Science,
Bangalore-560012, INDIA
vprasad@cedt.iisc.ernet.in

H S Jamadagni*
Ravi. A. Ravindranath+

+ MaXXan Systems
1975 Concourse Drive
San Jose, CA 95131

Abstract[1]: Real-Time services are traditionally supported on circuit switched network. However, there is a need to port these services on packet switched network. Architecture for audio conferencing application over the Internet in the light of ITU-T H.323 recommendations is considered. In a conference, considering packets only from a set of selected clients can reduce speech quality degradation because mixing packets from all clients can lead to lack of speech clarity. A distributed algorithm and architecture for selecting clients for mixing is suggested here based on a new quantifier of the voice activity called "Loudness Number" (LN). The proposed system distributes the computation load and reduces the load on client terminals. The highlights of this architecture are scalability, bandwidth saving and speech quality enhancement. Client selection for playing out tries to mimic a physical conference where the most vocal participants attract more attention. The contributions of the paper are expected to aid H.323 recommendations implementations for Multipoint Processors (MP). A working prototype based on the proposed architecture is already functional.

1. INTRODUCTION

Today's Internet uses the IP suite, which was primarily designed for transport of data. It offers best effort data delivery. Increasingly however, the Internet is being used as a transport mechanism for voice and video, which have different characteristics and requirements from those of traditional data. For "telephone conversation-type" applications, there are strict requirements on end-to-end delay and delay jitter. There are a few studies that examine the efficiency and quality of packetized voice [2]. It is found that due to statistical multiplexing and compression schemes, the efficiency in terms of the volume of voice traffic can be increased considerably. There are also some studies about the effects of CODEC (vide [7] for Codec standards) and QoS guarantees in [2] and [8].

The next step in the process of merging telephony with Internet is providing a number of facilities that a telephone network provides. Among them, the conference facility is the most important. Reasons for the popularity of audio and video conferencing on Internet are dealt with in detail in [1]; the advantages of audio and video conferencing have been thoroughly explored in [3] and [9]. A conference involves mixing audio streams from Clients in the conference to form a single stream and playing this stream at each client. In its simplest implementation, the bandwidth requirement for a conference over Internet is directly proportional to the number of clients. Reducing bandwidth for conferencing while maintaining audio quality is a challenge in Internet Telephony. Issues apart from bandwidth are: (a) packet delay, (b) echo, (c) mixing of audio from selected clients, (d) automatic selection of clients to participate in the conference, (e) playing of mixed audio at each client, (f) handling clients not capable of mixing audio streams (such clients are known as "dumb clients"), and (g) deciding the number of clients in conference without compromising on voice quality.

1.1 The Context and Requirements

In this paper, we address the problem of conferencing under a central "controller". Conferencing under central control is motivated by a corporate environment and is unlike the kind of conferencing where participants join and leave as they wish. In our scenario, participation is by invitation only. A conference is "built up" by adding participants successively; the addition can be done, for example, by the originator.

See Fig. 1 for an example. There are three major locations, each of which has an arbitrary number of clients.

There are two parts to the 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 programs (server programs) that facilitate conferencing. The requirement specifications are:

1. Registration: Prospective users (clients) should first "register" with a central control point to utilize/benefit from the conferencing software.
2. One-to-One call: A client should be able to call another client for 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 then there is a need to mix audio streams (say up to three or four). The conferencing software should provide this feature.
6. Consistency: Each client should get same set of audio streams for mixing to ensure that each client has the same view of the conference.

[1]This work was supported by Nortel Networks agreement number RIISOG9900HSJ.

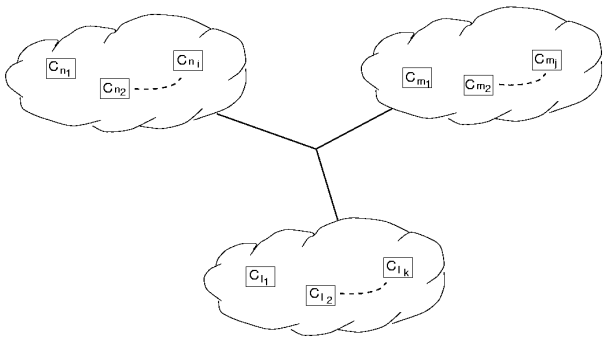

Fig. 1. A large number of clients dispersed over a wide geographical area.

7. Ease of addition/deletion of new clients: It should be simple to add/delete a client to/from an ongoing conference.
8. Scalability: The conferencing architecture should be scalable; it should be possible to handle a large number of clients dispersed over a wide geographical area.

Section 2 discusses the components of the architecture. Section 3 has a brief description of a new parameter called the Loudness Number, which is used for automatic selection of clients for mixing. Section 4 describes an algorithm and an example of automatic selection and switching of the audio streams. Implementation and conclusions are in Section 5.

2. ARCHITECTURE FOR CONFERENCING

2.1 Impact of requirements and proposed solution

Requirement 1 motivates a centralized control point called "Call Processor" (CP); this is like a Local Exchange in the telephone network. Requirements 2 and 3 imply that calls should be set up via the CP. Requirement 4 necessitates the use of IP multicast. As a conference is a one-to-many service, IP multicasting is a natural choice. In view of the possibly large number of clients (say K) and the limited amount of mixing required in practice, a client need not see voice packets from all other (K-1) clients. Limiting the number of packets reduces the processing burden for a client. This emphasizes the need for a component in the architecture that selects an appropriate number of packets for mixing. This component is called the "Selector", and it is a major step towards achieving scalability. Moreover there is a need for a proper selection criterion to choose the clients whose voice packets are to be mixed. This underlines the need for a metric that can be used to choose which clients' packets are to be mixed; this metric is the "Loudness Number".

As shown In Figure 1, clients may be distributed over a wide geographical area. Thus, we require a ***distributed*** mechanism for choosing the clients whose audio packets are to be mixed. In our architecture, each Selector serves a group of clients. For example, in Figure 1, there could be one Selector for each of the three groups. The Selectors communicate among themselves to arrive at a unique set of clients whose audio streams are to be mixed. The basis for forming the groups may be an optimisation algorithm that reduces the network traffic and computation load on Selectors.

As each Selector in a group restricts the number of packets selected for playout from that group and this remains constant, the addition of a client does not cause an increase in network activity outside that group.

2.2 Architecture

Our architecture is based on H.323 recommendations. The H.323 [5] recommendations are comprehensive, flexible, and can be applied to voice-only handsets as well as for full multimedia video-conferencing stations. H.323 defines the Multipoint Control Unit (MCU) and Clients/Terminals, which are the key elements in the architecture for conferencing support. The MCU is an endpoint on the network, which provides the capability for three or more Terminals and Gateways to participate in a multipoint

The MCU consists of a mandatory Multipoint Controller (MC), and optional Multipoint Processors (MP). The MC performs H.245 [6] multipoint control functions for a multipoint conference. Communication between the MC and the MP [11] is not subject to standardization.

Our proposed architecture for conferencing is shown in Fig. 2 and Fig 3. Functionally, the Call Processor (CP) is similar to the MC and the Selector is similar to the MP of H.323 document. All Selectors and clients are required to register with the CP.

The CP implements all the control messaging required for call set up and control. Fig. 3 shows voice flow between the clients and Selectors and between Selectors. The CP assigns a client to a particular Selector and forms a Selector group as in Fig. 2. The CP informs the corresponding Selector when a client joins the conference. The Selector then prepares to serve the client that has joined the conference. If the arrival of a new client brings in a new Selector, in turn a new Selector group, then the CP informs all other existing Selectors. The CP also decides "N", the number of clients to be selected finally for play out amongst all the clients in a conference and communicates this to all the Selectors.

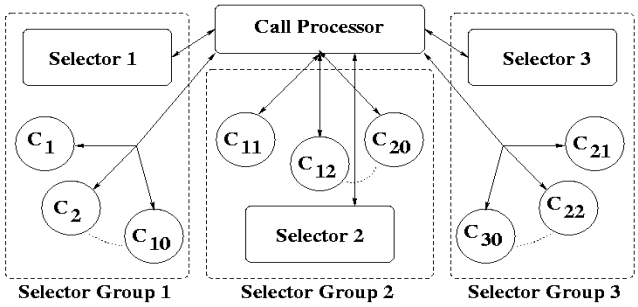

Fig. 2. Control Path Structure of VoIP Conference.

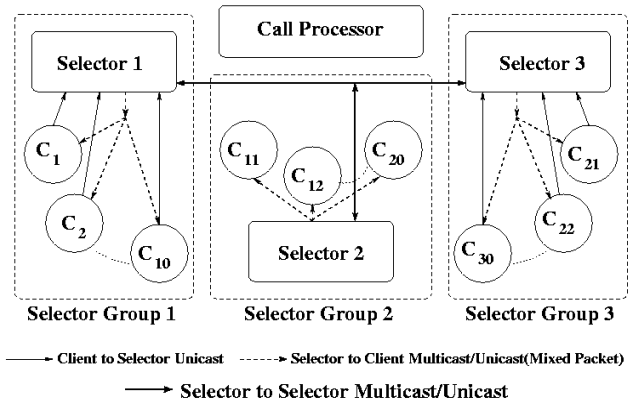

Fig. 3. "Data" (Voice) Path Structure of VoIP Conference.

The Selectors exchange packets with one another using IP multicast. In case a Selector is behind a router, which is not multicast-enabled, then unicast is used. As most LAN technologies support multicasting, Selectors normally talk to their clients on multicast. But in cases where clients cannot receive voice packets on multicast, the Selector has to send unicast packets to those clients as shown in Fig. 3.

Selectors also convert audio stream formats if necessary. After Selectors exchange audio data with one another, they form a global set of clients from which the audio packets are to be mixed and played out at the clients; we shall call this set *S*. Finally, a Selector sends all N packets from the set S to the clients in its domain along with their source identification (ID).

2.3 Working of a Selector

The operation of Selector 1 is shown in Figure 4. Selector 1 serves clients 1 to 10. For each mixing interval, Selector 1 chooses the "best" N audio packets out of the M1 it may possibly receive and sends these to Selectors 2 and 3. The set of packets sent is denoted as "*ToOtherSelectors*". In the same mixing interval, it also receives the best N audio packets (out of possibly M_2) from Selector 2, and the best N (out of possibly M_3) from Selector 3.[2] The set of packets received is denoted as "*FromOtherSelectors*". Finally, it selects the best N packets from the set {*ToOtherSelectors* ***union*** *FromOtherSelectors*} and passes these packets to its own group.

It can be seen that the set {*ToOtherSelectors* ***union*** *FromOtherSelectors*} is nothing but the set *S* in Section II.B, and it is the same at all Selectors. This ensures that any client in the conference finally receives the same set of packets for mixing. Hence all clients obtain a consistent view of the conference.

The "best" N packets are decided based on the "Loudness Number" (explained in Section 3) of each packet. Use of Loudness Numbers to select packets reflects the scenario in face-to-face conference wherein participants who talk louder and for longer time get noticed. Each client mixes the received packets with adjustable weights and the mixed audio stream is played out.

[2] For simplicity, we ignore propagation delay between Selectors.

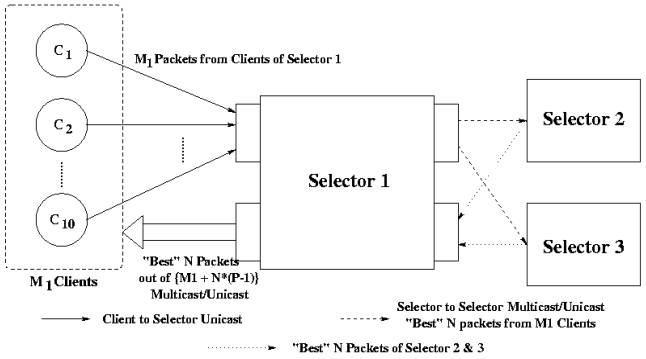

Fig. 4. Functional diagram of a Selector.

2.4 The features of the Architecture

1) Scalability: The bandwidth consumed in Selector-to-Selector communication is bounded above, because in every mixing interval, a Selector sends at most N audio packets to other Selectors, irrespective of the number of clients in its own group. The processing burden on a Selector is determined by the computation necessary to identify the set *ToOtherSelectors* in the "transmit" direction, and the set *S* = *{ToOtherSelectors* ***union*** *FromOtherSelectors}* in the "receive" direction. Addition of new clients to a conference results in a slight increase in computation (in the transmit direction) because the best N packets have to be chosen from a slightly bigger set. In the receive direction, the computation is insensitive to the addition of new clients, because in every mixing interval, the set *S* has at most N*P audio packets, where P = number of Selectors.

2) Enhancement in Quality of mixed audio: As Selectors send the identities (ID) of the N packets selected; it is possible to mix the packets at a client according to adjustable weights set by the user. This enhances the quality of the conference as each user can "tune" the mix to his/her liking.

3) Setting up of Multicast Tree: Selectors acts as a proxy for clients in its domain, therefore multicast tree can be built with only Selectors as nodes. The architecture avoids setting up of multicast tree whenever clients joins or leave the conference.

2.5 Performance aspects

It is clear from the discussion above that the architecture with Selectors leads to significant reduction in the processing requirements at clients, as well as communication costs on the wide area network. A small amount of extra processing is required at each client or at the Selectors in case of dumb clients to compute the Loudness Number (LN). The

complexity of LN calculation is shown to be of constant order [10].

3. THE LOUDNESS NUMBER

A basic problem to be solved by the Selector is: 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. However, this is usually found to be inadequate because random fluctuations in packet energies can lead to poor quality audio. This indicates the need for a metric different from mere individual packet energies. The metric should have the following characteristics:

- A person who is speaking (i.e., "has the floor") 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; this weight is often referred to as "Persistence" or "Hangover".
- By the same token, a participant who wants to interrupt the speaker will have to (i) raise his voice and (ii) 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.

The Loudness Number should quantify the following attributes of the speakers: (a) loudness of the speaker, (b) duration of the speaker's activity, and (c) level of activity for this duration. Therefore we define Loudness Number (LN) as a function of the energy of the audio stream, the duration of the past activity and level of activity for this duration. A detailed description of Loudness number characteristics and calculation is given in [10].

4. DISCUSSION

We consider briefly the format of the packets, which flow from a client to its Selector and from a Selector to its clients. The format is given in Fig. 5. The formats are a small extension of RTP [4]. We have proposed a few extra fields to support new facilities and this may be accommodated later into the existing RTP packet format. The packets from the clients have all the usual RTP fields plus a field for Loudness Number. But the packets from Selector to clients have sets of Contributing Source IDs, Loudness Number and the Data from the client. In case the packet is big it may be split into two or three chunks. An example of a packet from a Selector to its clients with N = 2 is shown in Fig. 5. The packetization interval is 40 ms in this example. Depending on whether 8 or 16 bit PCM encoding is used, the data length is 320 or 640 bytes, respectively. These packets are transmitted using UDP.

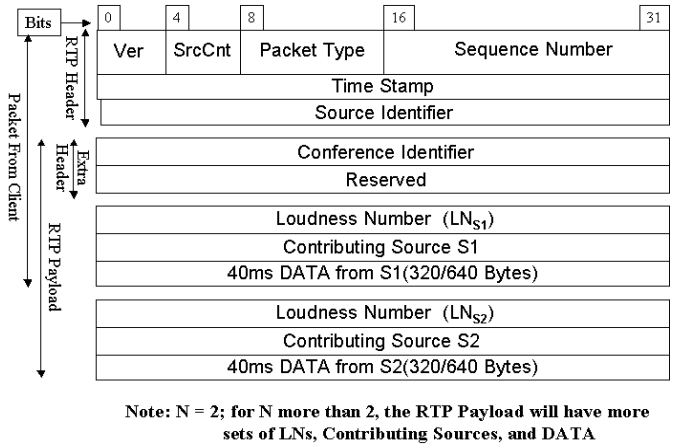

Fig. 5. UDP Payload Packet Structure (client to Selector, Selector to client and Selector to Selector).

4.1 The Algorithm

The algorithm is simple and self-explanatory. The algorithm run at each Selector is as follows:

Repeat for each time slot at each Selector
{

1. Get all the packets from the clients that belong to it.
2. Find at most N clients that have maximum LN out of M clients in its domain.
3. Store a copy of packets from those N clients thus selected in the Data Base (call it DB_1).
4. Send these N packets to other Selectors (on Multicast or Unicast as the case may be).
5. Receive similar packets from all other Selectors and store it in Data Base (say DB_2).
6. Now compare the packets in DB_1 and DB_2 on the basis of LN and select a maximum of N amongst them (call this as set S, having final N packets that should be played at each client).
7. Send these N packets in set S to the clients in its domain.
8. Mix these N audio packets in set S after linearising and send it to dumb clients in its domain.

}

4.2 An example

Let three Selector groups denoted by S_1, S_2 and S_3 in a conference. Let the value of N be 4. We shall take M = 10 for each Selector group as an example. We shall consider three sets at each Selector; namely, as Set *C* that contains the packets from each client in that Selector's domain, set *P* that contains the N packets selected from the set *C* on the basis of Loudness numbers and set *S* that contains N packets from all the Selectors, which will be played at all the clients in the conference.

In the notation used here, the Loudness Number is in bracket adjacent to the client, for example, *C : {1(34), 2(45)}* means the set *C* has two clients 1 and 2 and their respective Loudness Numbers are 34 and 45.

Now we shall take an example of the whole scenario of the conference and explain the working of the algorithm. The set *C* given below is for one time slot at any instant of time.

C_{S1} *: {1(80), 2(91), 3(22), 4(23), 5(24), 6(25), 7(35), 8(21), 9(20), 10(21)}*
C_{S2} *: {11(75), 12(55), 13(60), 14(21), 15(20), 16(21), 17(20), 18(21), 19(70), 20(21)}*
C_{S3} *: {21(95), 22(21), 23(44), 24(21), 25(50), 26(21), 27(20), 28(21), 29(40), 30(21)}*

Now each Selector will select the best N packets from its set *C* and form the set *P*. For this example the set *P* is,

P_{S1} *: {1(80), 2(91), 7(35), 6(25)}*
P_{S2} *: {11(75), 19(70), 13(60), 12(55)}*
P_{S3} *: {21(95), 25(50), 23(44), 29(40)}*

The Selectors will exchange packets in the sets P_{S1}, P_{S2} and P_{S3} among themselves. Finally, each of them finds out the N packets based again on Loudness Number amongst $P_{S1,}$ P_{S2} and P_{S3} to get the set *S: {2(91), 21(95), 1(80), 11(75)}*, calculated at each Selector independently would be same.

The packets from the set *S* will be mixed and played out at the clients with appropriate weights for each stream as selected by the client. In case of dumb terminals, the Selectors mix the packets in the set *S* with default weights and send a single packet to them.

Everyone in the conference, that is clients 1 to 30, will listen only to the clients in the set *S*: client number 2, 21, 1 and 11 during the current time slot. This is repeated for each time slot through the end of the conference. The set *C* will grow whenever a new client is allocated to a Selector group and will shrink when a client goes out of the conference. At a Selector, the cardinality of the set *P* might be less than N if the corresponding set *C* has less than N members.

5. IMPLEMENTATION AND CONCLUSIONS

The Call Processor and Selectors are implemented on Windows NT workstations. The clients are implemented on Windows 95 computers and the client program can run on any of the Windows 95/98/NT/2000 operating systems. A client has a graphical user interface for setting up the conference and changing the mixing weight of each client in the set *S*. The proprietary protocol between the CP and Selector is explained in [11]. Conferencing between multiple parties has been tested with this set up. There is a provision to set the value of N in the beginning of the conference.

This architecture avoids impulse sounds because of the way the loudness number is implemented. A persistent speaker gets into the conference. This set up models closely what happens in a typical face-to-face conference.

A design choice of N = 4 has worked well in our tests because in any normal conference, the number of persons speaking at any given instant of time will be only one, but in some cases when other members of the conference interrupt, it may be two, three or utmost four.

The facility of user-specifiable weights for mixing gives an opportunity for boosting the voice of the client whom the user wants to hear most clearly, as well as a simple way for a speaker to avoid listening to his own voice in the mix (echo suppression). This is achieved at the cost of Selectors sending N packets instead of a single mixed packet. However, given current LAN technology, the increased bandwidth requirement should pose no problem at all.

The Selectors can mix the N audio packets with a pre-defined weights and a single stream can be sent to dumb clients.

Some more facilities can be easily added: one of the clients can be the Moderator. In this case, the Moderator can be given "priority" by artificially increasing the Loudness Number of audio packets generated by him/her; this would ensure that the Moderator is always heard. The Loudness Number can be increased at the Selector serving the group to which the Moderator belongs, so that the client software need not be modified.

REFERENCES

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

[2] Mario Baldi and Fulvio Risso, "Efficiency of Packet Voice with Deterministic Delay", *IEEE Comm. Magazine*, May 2000, pp. 170-175.

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

[4] H. Schulzrinne et al., "RTP: a transport protocol for real-time applications", RFC 1889, IETF, Jan. 1996, ftp://ftp.isi.edu/in-notes/rfc1889.txt

[5] ITU-T Rec. H.323, "Packet based Multimedia Communications Systems", vol. 2, 1998, http://www.itu. int/itudoc/itu-t/rec/h/h323.html.

[6] ITU-T Rec. H.245, "Infrastructure of audiovisual services – Communication procedures", http://www.itu.int/itudoc/itut/rec/h/h245.html.

[7] Robert Shaw, reference implementation for CCITT G.711, G.721, G.723, Jun. 1993 http://www.itu.int /itudoc/itu-t/rec/g/g700-799/index.html.

[8] Markku Korpi and Vineet Kumar, "Supplimentary Services in the H.323 IP Telephony Network", *IEEE Comm. Magazine*, July 1999, pp. 118-125.

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

[10] R Venkatesha Prasad, Joy Kuri, H S Jamadagni, Haresh Dagale and Ravi Ravindranath, "Automatic Addition and Deletion of clients in VoIP Conferencing" *6th IEEE Symposium on Computers and Communications 2001*, Hammamet, Tunisia, pp. 386-390.

[11] R Venkatesha Prasad, Joy Kuri, H S Jamadagni, Haresh Dagale and Ravi Ravindranath, "Control Protocol for VoIP Audio Conferencing Support", *International Conference on Advanced Communication Technology*, Mu-Ju, South Korea, Feb 2001, pp. 419-424.