A Multiplexing Scheme for H.323 Voice-Over-IP Applications

H. P. Sze, Soung C. Liew, *Senior Member IEEE*, Jack Y. B. Lee, and Danny C. S. Yip

***Abstract*—Voice communications such as telephony are delay sensitive. Existing voice-over-IP (VoIP) applications transmit voice data in packets of very small size to minimize packetization delay, causing very inefficient use of network bandwidth. This paper proposes a multiplexing scheme for improving the bandwidth efficiency of existing VoIP applications. By installing a multiplexer in an H.323 proxy, voice packets from multiple sources are combined into one IP packet for transmission. A demultiplexer at the receiver-end proxy restores the original voice packets before delivering them to the end-user applications. Results show that the multiplexing scheme can increase bandwidth efficiency by as much as 300%. The multiplexing scheme is fully compatible with existing H.323-compliant VoIP applications and can be readily deployed.**

***Index Terms*—Internet telephony, multiplexing, voice over IP.**

I. Introduction

THE DEVELOPMENT and expansion of the Internet in the last few years are making it possible for real-time voice traffic to reach many different corners of the world. Although the Internet was not originally designed for real-time communications, Internet telephony is becoming one of the fastest-growing application areas for the Internet today [1]. With the deployment of quality-of-service (QoS)-capable network components, the primary hindrance to voice over IP (VoIP) can be overcome and its proliferation will be almost certain in the near future [2].

The tremendous growth of Internet telephony is driven by its several fundamental benefits over traditional public switched telephone network (PSTN). First of all, from the user's point of view, making long-distance calls via the Internet results in substantial cost reduction since international toll charge imposed by telecommunication companies are bypassed. In addition, voice communication can be enhanced with other media and data applications like video, white boarding, and file sharing. Some of these add-on features have already been implemented in software applications such as NetMeeting [3].

From the network operator's viewpoint, IP telephony can have two substantial bandwidth advantages. First, advanced voice-compression techniques [4], can be used to reduce the data rate of a call. A 64-Kb/s channel in a PSTN call can be used to accommodate several IP-based telephony calls, which consumes less than 10 Kb/s each with a slight degradation in voice quality. Second, voice delivery over IP network itself can achieve higher efficiency as compared with PSTN. Specifically, with circuit switching in PSTN, a dedicated channel is reserved over the whole duration of a call, whereas with the packet-switched Internet, bandwidth is consumed only when voice packets are delivered.

In spite of the advantages, voice over the Internet has several lingering challenges. Apart from the potential QoS problem, one key issue is the inefficient use of bandwidth in high-cost wide area network (WAN), namely the access links of enterprise network, due to packet header overhead (although efficiency is already much better than the use of PSTN). For example, a typical IP voice packet consists of a header of 40 bytes. Compared with the typical payload size of only 10–30 bytes for each audio frame, the header overhead is clearly very substantial.

Current VoIP applications tackle this problem by embedding multiple audio frames into a single packet at the source to increase the ratio of payload to header size [6]. This approach has the benefit of reducing the overall data rate of a call. However, since an audio frame is generated only after raw audio signals in a frame period are captured and encoded, packing an additional audio frame will add another frame period to the assembly delay. Together with the existing network delay, the resultant end-to-end delays may become unacceptable.

Obviously, maintaining a short delivery time of data is essential in interactive applications like telephony. In particular, the ITU Recommendation G.114 [7] gives guidelines on the tolerable delay for a normal telephone conversation. The maximum one-way end-to-end delay acceptable by most users is only 150 ms. A telephony system with longer end-to-end delays will cause users to engage in collided talks and mutual silences more often [8]. To balance conversation quality and network efficiency, a compromise must be made between packet delay and header overhead. Such a tradeoff is illustrated by Fig. 1, in which a G.723.1 codec is assumed.

In this paper, we present a novel voice multiplexing scheme which makes it possible to circumvent the above tradeoff to achieve low end-to-end delay and high bandwidth efficiency simultaneously. In particular, our solution aims for bandwidth efficiency on a shared long-distance link used to transport the

Manuscript received May 1, 2001; revised January 7, 2002. This work was supported in part by the Area of Excellence in Information Technology of Hong Kong: Project AoE/E-01/99.

H. P. Sze was with the Department of Information Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong and a Consultant with Roctec Technology Ltd. He is now with Hong Kong Applied Science and Technology Research Institute, Kowloon, Hong Kong (e-mail: sze@astri.org).

S. C. Liew and J. Y. B. Lee are with the Department of Information Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong (e-mail: soung@ie.cuhk.edu.hk; jacklee@computer.org).

D. C. S. Yip was with the Department of Information Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong. He is now a Consultant with Accenture Company Limited, Hong Kong (e-mail: dyip@hongkong.com).

Publisher Item Identifier 10.1109/JSAC.2002.802064.

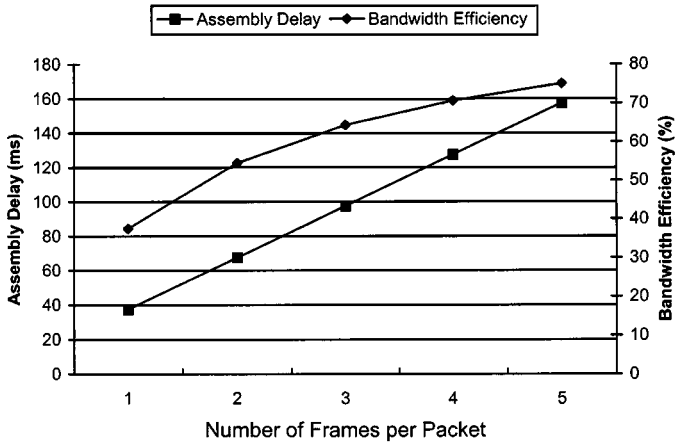

Fig. 1. Tradeoff between bandwidth efficiency and assembly delay with G.723.1.

voice packets. Audio frames from various sources are transmitted on a shared IP packet in our scheme to reduce header overhead.

II. Background

In this section, we review current VoIP standards that are relevant to our work, including speech codec, H.323 communications protocol, and RTP/UDP/IP header compression.

A. Speech Codec

The primary functions of a speech codec are to perform analog/digital voice signal conversion and digital compression. Three commonly used codecs in Internet telephony are GSM 6.10 [9], G.723.1 [10], and G.729A [11]. Table I summarizes the main attributes of these codecs. One observation is that all three codecs generate audio frames at a constant bit-rate. When silence suppression scheme is employed, the codecs then operate in two states: a silent state at zero bit-rate (or lower bit-rate in some other codecs) and an active state at the compressed bit-rate. Regardless of the state, the frame period and frame size are still fixed. This characteristic allows us to simplify the design of our multiplexing scheme, to be discussed in Section III-B.

B. The H.323 Standard

Two major standards are emerging in the industry for VoIP: the ITU-T Recommendation H.323 [12] and the session initiation protocol (SIP) [13] from Internet Engineering Task Force (IETF). Being the earlier of the two standards, H.323 has received wider adoption in the industry since its first release in 1996. VoIP product vendors such as Cisco Systems, Microsoft and VocalTec all support this standard in their VoIP applications. For this reason, we design our multiplexing scheme to be compatible with the H.323 standard so that existing H.323-compliant applications can be seamlessly supported.

H.323 is known as an umbrella recommendation as it refers to other standards in its system description. Its main specifications include call signaling, media-channel signaling, and media transportation. It also introduces an important component where we implement our system—the gatekeeper. Gatekeeper is an H.323 entity on the network that provides address translation, access control for terminals, and optionally, call routing for signaling and control messages. All these features are essential to our scheme and are adopted in the call-establishment procedures of the multiplexing system.

C. RTP/UDP/IP Header Compression

Real-time transport protocol (RTP) provides end-to-end network transport functions needed for applications transmitting real-time data [14]. It is now being used by various network applications, including H.323, for delivery of media stream. Since RTP packets transmitted over the Internet carry a 40-byte RTP/UDP/IP header, the header overhead will become very substantial for packets with small payload size (e.g., 10 bytes in G.729A at 1 frame/packet). Thus, to increase the bandwidth utilization and minimize the end-to-end packet delay, Casner and Jacobson have proposed an RTP/UDP/IP header compression scheme for low-speed serial links [15]. Their work has successfully reduced the RTP/UDP/IP header size from 40 bytes to a minimum of 2 bytes on a link-by-link basis. Based on similar principles, we designed a similar but simplified compression scheme for use in the RTP application layer (as opposed to the link layer in the original case) to minimize the header length of audio chunks from different sources inside a multiplexed packet so as to increase the bandwidth efficiency.

D. Related Work

The idea of multiplexing multiple voice streams has also been investigated by other researchers. Hoshi [16] proposed a multiplexing system for use between H.323 IP-telephony gateways to reduce the number of RTP packets transported over the IP network. His multiplexer improves transmission efficiency by concatenating voice packets from different streams into a single UDP packet for transmission. But unlike the multiplexing scheme studied in this paper, their scheme did not compress the RTP headers.

More recently, IETF is working on a multiplexing scheme for RTP streams [17]. Rather than integrating voice-multiplexing technique with the existing VoIP systems, their work combined the aforementioned RTP/UDP/IP compression standard with another point-to-point protocol (PPP) multiplexing scheme [18] to form a generic method for end-to-end tunneling of multiplexed RTP packets. Their proposed scheme operates in the link layer and can only be used over PPP links.

III. VoIP Multiplexing Scheme

We present in this section the design of the proposed VoIP multiplexing scheme and discuss how it can be integrated into the H.323 framework to maintain compatibility with existing H.323 applications. The design of the multiplexer is motivated by two observations. First, bandwidth at a local network [e.g., corporate local area network (LAN)] is usually abundant and, hence, efficient bandwidth usage is not a major concern. Second, multiple local networks are often connected via one or more WAN links with more expensive and limited bandwidth. These WAN links are the bottleneck and, hence, bandwidth efficiency is critical. The proposed multiplexer addresses this challenge by

TABLE I
THE ATTRIBUTES OF THREE COMMON SPEECH CODECS

Codec	GSM 6.10	G.723.1	G.729A
Bitrate (Kb/s)	13	5.3 / 6.3	8
Frame size (bytes)	32.5	20 / 24	10
Frame period (ms)	20	30	10
Lookahead delay (ms)	0	7.5	5
Total encoding delay (ms)	20	37.5	15
Decoding delay (ms)	20	18.75	7.5
Quality	Fair	Fair	Good

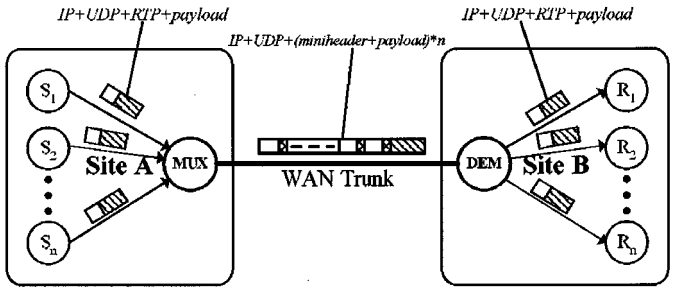

Fig. 2. System architecture.

multiplexing and compressing voice packets before transmitting them over the WAN links. The following sections present the multiplexing algorithm, the header compression scheme, and the connection management protocol.

A. Packet Multiplexing

As bandwidth in local networks is abundant, VoIP applications can send each audio frame in a separate RTP packet to minimize packetization delay. For voice traffic to be transported over the same WAN link, they go through a multiplexer (MUX) located at the sender-side network, and a demultiplexer (DEMUX) located at the receiver-side network. The MUX replaces the RTP header of each voice packet with a compressed miniheader and then aggregates multiple packets into a single multiplexed packet. All these chunks share a common UDP/IP header and are then sent to the DEMUX across the WAN link. On the receiver-side network, the DEMUX inspects the miniheaders inside each multiplexed packet, restores the original RTP headers and destination information, and reconstructs the individual voice packets for delivery to their respective receivers. The system architecture is illustrated by Fig. 2.

To control packet delay, the MUX sends out a multiplexed packet every T ms, which is equal to or shorter than the audio-frame period, denoted by k ms. Larger value of T can improve bandwidth efficiency for more streams can be multiplexed but the delay incurred will also be larger. Hence, by adjusting T, one can control the tradeoff between bandwidth efficiency and delay. In addition, by setting $T \leq \min\{k_i\}$, where k_i is the frame period of the ith source, the multiplexing system can also be applied to situations where senders are using speech codecs with different frame periods.

The previous multiplexing policy ensures that the extra delay incurred in the MUX is bounded by one frame period. Moreover, since at most one audio frame from each source is stored in an aggregated packet, the design of our scheme is simplified by the limitation of only one possible payload size in each chunk of a source embedded in the aggregated packets. Alternatively, the MUX may also generate a multiplexed packet before T ms expires if the amount of accumulated payload has reached the maximum transfer unit (MTU) of the link layer of the WAN trunk. This can avoid increasing the packet-loss probability due to fragmentation of the multiplexed packets in the WAN link.

B. RTP Header Compression

Apart from aggregation, we also increase the bandwidth efficiency of multiplexed voice traffic by compressing the packet headers during multiplexing. As mentioned in the previous section, the compression algorithm in our scheme draws upon the ideas of RTP/UDP/IP header compression described in [15], with features optimized for VoIP applications, discussed below.

Casner's algorithm takes advantage of two properties in most types of RTP streams. The first is that most of the fields in the IP, UDP and RTP headers do not change over the lifetime of an RTP session. These constant-value fields can be represented by fewer bits with a session context during transmission. Second, RTP header fields like sequence number and timestamp are increased by a constant amount for successive packets in a stream. Hence, differential coding can be applied to compress these fields into few bits.

Combining the previous techniques, Casner managed to compress the header size down to two bytes in the best case (one condition, e.g., is that UDP checksum from the source is disabled) but the scheme relies on the link layer for exchanging control messages. While this scheme offers a full restoration of the RTP/UDP/IP header, we incorporate a simplified version of it into our application-layer scheme to suit voice traffic and recover only the necessary header fields for lossless playback of voice data at receiver. In this way, 2-byte header compression can be achieved for most voice packets in a RTP stream without the support of the link layer, making it possible to apply our scheme at the network layer in which packets may traverse several links.

As long as the packets can be transported to the correct destination, the fields in the UDP/IP header which do not need to

TABLE II
DIFFERENT ATTRIBUTES IN MUX/DEMUX TABLES

Primary key for MUX to identify DEMUX IP & port — *Primary key for DEMUX to identify Channel ID*

Channel ID	MUX IP	DEMUX IP	DEMUX Port

(a) Channel table in MUX/DEMUX.

Primary Key

Source		RTP	CID	Payload	Active	Idle	Last Synchronization		
IP	Port	SSRC		Type	Size	Size	CSEQ	RTP Seq. No.	Timestamp

(b) Context-mapping table for a particular channel ID in MUX.

Primary Key

CID	Destination		Active	Idle	Time	RTP	Last Synchronization		
	IP	Port	Size	Size	Difference	Header	CSEQ	RTP Seq. No.	Timestamp

(c) Context-mapping table for a particular channel ID in DEMUX.

be recovered are source IP, port number, IP packet ID and UDP checksum. For successful delivery, two important elements to be reconstructed during decompression in our scheme are destination IP address and port number. The fields in the RTP header are all reconstructed to preserve the timing information of the audio payloads.

Our compression mechanism depends on the use of context-mapping tables in both MUX and DEMUX. These tables record necessary information for proper (de)compression and (de)multiplexing. The different attributes in the MUX/DEMUX tables are shown in Table II. Since a MUX can simultaneously work with multiple DEMUX's and vice versa, a particular multiplexed channel is identified by the combination of MUX IP, DEMUX IP and DEMUX receiving port number. Each channel has a unique ID and context-mapping table. In addition, with the 1-byte-long session context identifier (CID), each channel can accommodate at most 256 sessions. Thus, using several DEMUX ports, multiple channel IDs can be assigned to a MUX/DEMUX pair to increase the number of sessions by handling several virtual multiplexed channels on the same physical channel.

Three types of miniheaders are used in the compression. COMPRESSED is a fully compressed RTP header. SYNCHRONIZATION is a compressed RTP header with sequence number and timestamp. UNCOMPRESSED contains a full RTP header. Their formats are shown in Fig. 3.

The header compression process is as follows. When a call is made between two regions, an entry with a unique CID is inserted into the tables in both the MUX and DEMUX during the H.323 connection establishment stage, which will be detailed later in this paper. Afterwards, the MUX will begin to receive voice packets in the RTP session from this source. Upon receiving a packet, the MUX will first search for a matched session context by the primary key—source IP address, port number, and RTP SSRC identifier inside each channel's context-mapping table until a correct multiplexed channel (or DEMUX) and CID can be found. For the first few

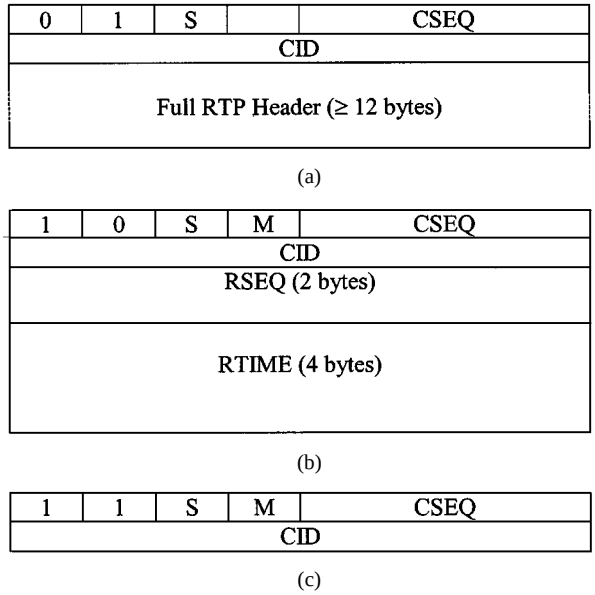

Fig. 3. Formats of three types of miniheaders. (a) UNCOMPRESSED miniheader. (b) SYNCHRONIZATION miniheader. (c) COMPRESSED miniheader.

RTP packets from the source, they will be embedded in the multiplexed packets with the UNCOMPRESSED miniheader (without RTP header compression). This is for the DEMUX to fill in the fields in its context-mapping table like RTP Header, Time Difference, and Last Synchronization. After that, if the RTP header fields in subsequent packets can fulfill the criteria of compression (i.e., constant changes in the timestamp and sequence number, and no changes in the constant RTP fields), a 2-byte COMPRESSED miniheader, which consists of a CID and context sequence number (CSEQ), will replace the 12-byte RTP header at the MUX. The CSEQ is calculated from the difference between the RTP sequence number in the received packet and the one updated in the MUX table entry in the previous MUX/DEMUX synchronization. On the demultiplexing side, by checking the receiving port and the CID of a voice

TABLE III
AN EXAMPLE OF CONTEXT MAPPING TABLES

Source		RTP	CID	Payload	Active	Idle	Last Synchronization		
IP	Port	SSRC		Type	Size	Size	CSEQ	RTP Seq. No.	Timestamp
IP_A	$Port_A$	X	10	PT	10	0	2	40	14200

(a) Example of a context-mapping table in MUX.

CID	Destination		Active	Idle	Time	RTP	Last Synchronization		
	IP	Port	Size	Size	Difference	Header	CSEQ	RTP Seq. No.	Timestamp
10	IP_B	$Port_B$	10	0	10	...	2	40	14200

(b) Example of a context-mapping table in DEMUX.

chunk embedded in a multiplexed packet, a DEMUX can look up a matched entry in its tables. It then reconstructs the constant RTP fields from the stored RTP Header attribute. To recover the changing RTP fields, it adds multiple of the first-order differences to the saved RTP sequence number and timestamp during last synchronization according to the CSEQ inside the miniheader.

For better illustration, we demonstrate this idea by an example. Assume at a particular moment, the context mapping tables in MUX and DEMUX contain the information as shown in Table III.

Now, suppose the MUX receives an RTP packet with sequence number and timestamp equal to 43 and 14 230, respectively. It computes the CSEQ for the COMPRESSED miniheader as follows:

$$\begin{aligned}\text{CSEQ} &= 43 - \text{Last Sync. RTP Seq. No.} \\ &\quad + \text{Last Sync. CSEQ} \\ &= 43 - 40 + 2 \\ &= 5.\end{aligned}$$

As a result, the MUX embeds the voice chunk stamped with CSEQ $= 5$ into its multiplexed packet. When this packet arrives at the DEMUX, a record is found for this payload and the original varying RTP fields is regenerated by the following computations:

$$\begin{aligned}\text{RTP Seq. No.} &= \text{CSEQ} - \text{Last Sync. CSEQ} \\ &\quad + \text{Last Sync. RTP Seq. No.} \\ &= 5 - 2 + 40 \\ &= 43 \\ \text{Timestamp} &= (\text{CSEQ} - \text{Last Sync. CSEQ}) \\ &\quad \times \text{Time Difference} + \text{Last Sync.} \\ &\quad \text{Timestamp} \\ &= (5 - 2) \times 10 + 14\,200 \\ &= 14\,230.\end{aligned}$$

Other constant RTP fields can also be recovered since they have already been stored in the DEMUX table. Nevertheless, when the MUX receives a packet with the second-order difference of RTP timestamp not equal to zero (e.g., the packet received at the start of a talkspurt with a codec that outputs zero bit during silent state), the DEMUX cannot restore the original timestamp from the COMPRESSED miniheader. Accordingly, a synchronization miniheader, which carries the RTP sequence number and timestamp, will be sent instead. Furthermore, since the four-bit CSEQ in a miniheader produces a sequence-number cycle of 16 packets, this synchronization miniheader will also be transmitted by the MUX once every 16 packets for each RTP stream.

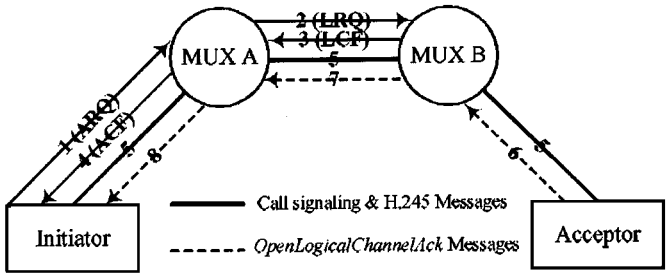

Fig. 4. Connection procedures.

In our miniheaders, there are one State (S) bit and one Marker (M) bit. With codecs operating with two-state variable bit-rates, the DEMUX can derive the bit-rate of an RTP stream from the S bit and, hence, the payload size after the miniheader of a particular voice chunk in a multiplexed packet. The M bit is an RTP field that is set to one in the first packet of a talkspurt. This bit must be carried to the DEMUX for complete restoration of the RTP header.

C. Connection Establishment

The H.323 standard specifies the process of connection setup in a phone call over a packet network. The main phases include registration/admission signaling (RAS), Q.931 call signaling, and H.245 media channel signaling. To ensure that any H.323 clients can readily use our multiplexing system to make calls without modification, we embed our connection establishment procedures for multiplexing into the H.323 calling process. For seamless incorporation, our MUX/DEMUX will act as an H.323 gatekeeper as well. All the steps required for establishing the multiplexing channel are transparent to the clients. Depicted by Fig. 4, the whole connection procedures are as follows.

1) The H.323 initiator registers at MUX A and sends it an *Admission Request* (ARQ) carrying the alias of the H.323 acceptor.
2) MUX A searches for the acceptor by broadcasting a *Location Request* (LRQ) to its MUX neighborhood, including MUX B.

3) Assuming the called party is currently logging into MUX B, MUX B will reply MUX A with a *Location Confirm* (LCF). For routing the signaling messages through MUX B, its LCF carries its own Call Signaling Channel Transport Address instead of the original called party.
4) MUX A sends an *Admission Confirm* (ACF) to the initiator, with the Transport Address set to itself instead of MUX B.
5) The endpoints exchange call signaling and H.245 control messages through the MUXs.
6) However, when MUX B receives the *OpenLogicalChannelAck* (ACK) message from the acceptor, which contains this endpoint's destination IP address and port number for the RTP stream from the initiator, MUX B stores these two fields in a new entry of its DEMUX table and replaces them in the ACK with its IP address and the receiving port of this multiplexed channel. Meanwhile, a unique CID is assigned to the RTP stream and is embedded into an extra and optional *portNumber* field in the ACK, which is not used originally. This message is then forwarded to MUX A.
7) When MUX A receives this modified message, it also creates a new entry in its MUX table with the specified CID. After that, it records the DEMUX IP and receiving port in the channel table if needed and replaces them in the ACK with its IP address and receiving port for the RTP source.
8) The initiator receives the ACK from MUX A and starts transmitting voice packets to it with the given IP address and port number. [Step 6)–8) will be repeated in the opposite direction when the initiator feeds back an ACK to the acceptor through the MUXs for establishing an RTP session from the acceptor to initiator.]

After successful connection setup, the first few voice packets from this particular source will not be compressed when it is being multiplexed with the voice chunks from other sources so that the remaining fields in the DEMUX table entry of MUX B can be filled in. After that, MUX A will then compress and multiplex the RTP packets headed for the acceptor by maintaining the CID and the multiplexed channel negotiated during startup.

IV. Performance Analysis

In this section, we analyze the performance of the proposed multiplexing scheme using three performance metrics, namely bandwidth efficiency, capacity, and delay.

A. Bandwidth Efficiency

We first compare the bandwidth efficiency in the IP layer of the proposed multiplexing scheme to the conventional nonmultiplexed approach. The efficiency of a WAN link connecting the proposed MUX and DEMUX can be calculated from

$$\rho_{\mathrm{MUX}} = \frac{\text{payload size}}{\text{IP packet size}} = \frac{a \times n}{h_{\mathrm{UDP}} + (h_m + a) \times n} = \frac{a}{\left(\frac{h_{\mathrm{UDP}}}{n} + h_m\right) + a} \tag{1}$$

where a is the audio-payload size, h_m is the COMPRESSED miniheader size (2 bytes), h_{UDP} is the UDP/IP header size (28 bytes), and n is the number of voice streams multiplexed.[1]

For the conventional method without multiplexing, the transmission efficiency is given by

$$\rho_{old} = \frac{\text{payload size}}{\text{IP packet size}} = \frac{a}{h_{\mathrm{RTP}} + a} \tag{2}$$

where h_{RTP} is RTP/UDP/IP header size (40 bytes).

As an illustration, take the common G.729A codec which generates voice packets at a rate of 8 Kb/s as an example. The audio-frame size is equal to 10 bytes and, hence, if each RTP packet from the source carries one frame, we get $\rho_{\mathrm{old}} = 0.2$. In other words, 80% of the bandwidth is used for delivering headers instead of voice data.

Using our multiplexing scheme, suppose there are 45 simultaneous calls in active state (i.e., sending voice data), then the efficiency ρ_{MUX} becomes 0.792. Comparing to the nonmultiplexed approach, our multiplexing scheme increased bandwidth efficiency by almost 300%.

B. Capacity

Knowing the bandwidth requirement and available channel capacity, we can then compute the maximum number of voice streams that can be accommodated. Assuming the capacity is limited by available bandwidth, then we can determine the numbers of concurrent streams using the following inequality:

$$\text{Aggregated voice data rate} < \text{available bandwidth}. \tag{3}$$

Suppose all transmitters use the same codec with a frame period of k ms. For the proposed scheme, the multiplexing period will be equal to one frame period, and the inequality becomes

$$((h_m + f) \times n + h_{\mathrm{UDP}}) \times 8000/k < B \tag{4}$$

where f is the audio-frame size and B is the WAN-link bandwidth in bps. Without multiplexing, the inequality becomes

$$(h_{\mathrm{RTP}} + (r \times f)) \times n \times 8000/(r \times k) < B \tag{5}$$

where r is the audio frame-to-packet ratio.

Again, using G.729A for comparison (i.e., $f = 10$ and $k = 10$). Consider a T1 link of 1.5-Mb/s bandwidth. Then our multiplexing scheme can accommodate up to 153 streams according to (4). By contrast, the nonmultiplexed approach can accommodate only 37 streams with one audio frame per RTP packet. With two audio frames per RTP packet, the number of streams increases to 62, which is still only 40% of the capacity achieved by our multiplexing scheme.

C. Delay

The tradeoff in achieving higher bandwidth efficiency using multiplexing is delay. Specifically, two types of delay are incurred by the multiplexing scheme: additional packet processing time at the router and packet waiting time at the MUX. Since

[1]The effect of periodical transmission of SYNCHRONIZATION miniheader on efficiency is insignificant and is omitted in the derivations.

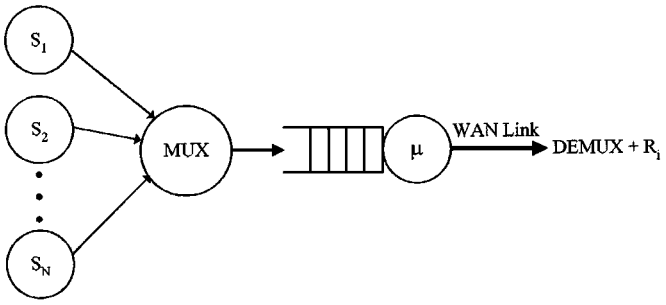

Fig. 5. Network scenario in simulations.

the router store-and-forward delay is proportional to the packet length and inversely proportional to the link speed, the larger multiplexed packets will introduce a longer transmission delay than normal voice packets. But because the multiplexed-packet length is bounded by the MTU, this store-and-forward delay is at most 3 ms for a typical T1-speed link with MTU $= 576$ bytes.

A voice packet will also encounter an extra delay at the MUX when it needs to wait a certain amount of time before the MUX generates a multiplexed packet. Since we set the multiplexing period to one audio-frame period and the MUX may alternatively send a packet when the number of voice chunks collected is adequate for transmission, this delay is bounded by one frame period. Because senders need not worry about the bandwidth efficiency at the WAN trunk with the multiplexing feature, they can simply transmit one audio frame in every RTP packet to minimize the overall packet latency. As a result, the delay incurred at the source is one frame period. Since the processing time of MUX and DEMUX is insignificant, the total delay introduced by our system is limited by about two frame periods. This amount of latency is close to the conventional scheme transmitting 2 frame/packet at the source but our scheme can offer a much higher bandwidth efficiency when there are many voice streams.

V. Simulation Results

We carry out simulations to study the performance of our system. Our goal is to compare the results with the conventional schemes in terms of packet delay and number of calls supported.

A. Network and Source Model

The network model in our simulations consists of two localities connected together with a WAN link of limited bandwidth, one containing N senders and the other N receivers. The senders S_i are assumed to connect to a common MUX via an infinitely fast link in a LAN. The receivers R_i are assumed to connect to a common DEMUX via an infinitely fast link in another LAN. The WAN trunk is modeled as a first-in first-out (FIFO) queue with service rate $= \mu$ and a fixed buffer size. Incoming packets are discarded when the buffer overflows. The scenario is shown in Fig. 5. The MUX multiplexes the RTP packets from senders every T ms which is equal to one audio-frame period or when the multiplexed packet is full according to the link layer MTU of the WAN which is 576 bytes in our simulations. Consequently, the MUX introduces a variable delay to every packet from senders. The propagation times and processing times of MUX/DEMUX and receivers are assumed to be negligible.

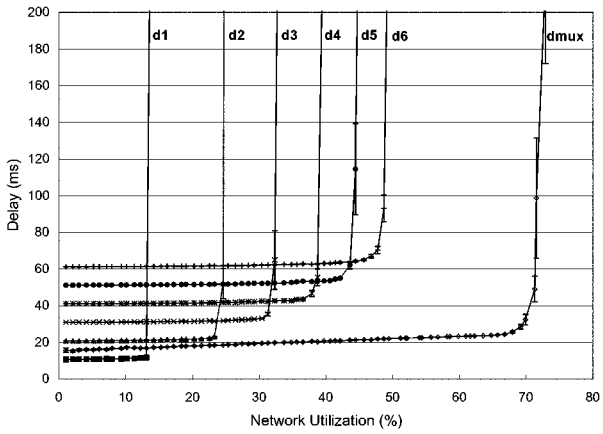

Fig. 6. Comparison of overall delay and WAN link utilization between conventional schemes and proposed scheme.

We assume that voice silence suppression is activated in the codecs and, therefore, each voice source in the simulations can be modeled as a simple two-state variable-bit-rate ON–OFF source, a traffic model extensively used in standard voice sources [19]. This source generates audio frames of size D_{ON} and D_{OFF} in the ON (active) and OFF (idle) state, respectively. We let the ON and OFF state durations be distributed exponentially with means $1/\alpha = 352$ ms and $1/\beta = 650$ ms, which are commonly used values in modeling phone calls [20]. The frame-generating instants of different senders are assumed to be evenly distributed with one frame period. For the proposed scheme, an RTP packet is transmitted once a frame is produced.

B. Packet Delay Versus Network Utilization

Fig. 6 depicts the average overall packet delay, which includes assembly delay, sojourn time at the WAN link and waiting time at MUX if it is used, at different WAN link utilization. We set the WAN trunk bandwidth to 512 Kb/s and vary the network utilization by increasing the number of simultaneous connections N. G.729A codec is assumed so that $T = 10$ ms, $D_{\mathrm{ON}} = 10$ bytes, and $D_{\mathrm{OFF}} = 0$ bytes. The utilization is obtained by the payload data rate over the link capacity. Curves d1–d6 show the performance of conventional schemes with one to six audio frames packed into an RTP packet. Curve dmux gives the overall delay incurred by our multiplexing scheme. 90% confidence intervals are shown for each point.

Comparing the curves d1 and dmux, we observe that the delay performance of our scheme is comparable to the conventional scheme of one audio frame per packet. However, our scheme outperforms the conventional scheme in terms of network utilization. When using the conventional scheme d1, network utilization can only be pushed up to 13%. But with our technique, a network utilization of as high as 72% is attained, corresponding to a 450% improvement. Curve d2 also has similar delay performance to our scheme. Although its network utilization is improved to 25%, ours is still almost three times of it. Adopting 6 frame/packet in the conventional scheme (d6) can achieve a

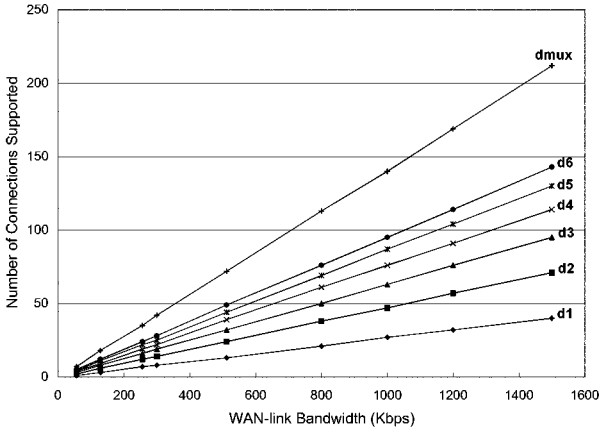

Fig. 7. Number of supported connections of different schemes over WAN links of various bandwidths.

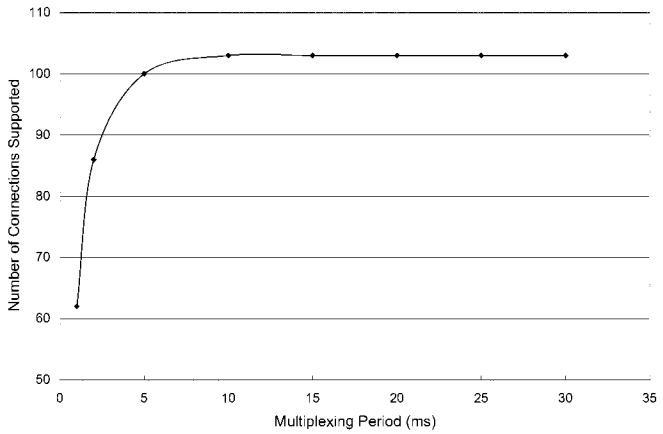

Fig. 8. Number of connections supported at various multiplexing periods.

utilization of 49%; however, a delay of 60 ms is incurred and its efficiency is still far from our system.

C. Number of Connections Supported

Next, we investigate the number of connections sustainable by the WAN link at various bandwidths from 64 Kb/s to 1.5 Mb/s. In each simulation, we again vary the link utilization by increasing the number of simultaneous connected calls N. When the delay variance jumps drastically from around 10 ms^2 to more than 200 ms^2, this means that the link is overloaded. Therefore, the number of supported calls for each link bandwidth $N_{\max}$ can be collected just before the delay variance at $(N_{\max}+1)$ connections becomes significant (larger than 200 ms^2).

It can be seen from Fig. 7 that our proposed scheme can be applied to WAN links of different bandwidths without affecting its efficiency. This shows that the scheme always outperforms the conventional schemes in terms of number of supported connections. Specifically, it can support about 200% more connections than the 2 frame/packet conventional scheme, which is of similar delay performance, in the range of tested link bandwidths. In general, the difference in number of sustainable connections between a conventional scheme and our scheme increases as the WAN-trunk bandwidth increases.

D. Multiplexing Period Versus Number of Connections Supported

Finally, we study how the selection of a multiplexing period in our system affects the number of sustainable calls. In our design, this period can freely be set to a value not greater than the frame period. Clearly, a shorter period reduces the multiplexing delay but increases the header overhead because each multiplexed packet aggregates fewer voice streams.

For this simulation, we assume G.723.1 to be the codec used by the sources. Therefore, the frame period $T = 30$ ms, $D_{\mathrm{ON}} =$ 24 bytes and $D_{\mathrm{OFF}} = 0$ bytes. Fig. 8 shows the number of calls supported by a 512-Kb/s WAN trunk with multiplexing period ranging from 1 to 30 ms. We notice that the number of calls increases quite significantly when the period is varied from 1 to 10 ms.

Knowing this relationship, it is possible for a MUX operator to adjust the multiplexing period according to the number of existing connections in the MUX so that the system can support all the voice calls with the smallest possible delay to attain the best-possible conversation quality. For example, to support 100 calls only, it is safe enough to set the period to 5 ms to reduce unnecessary delay.

It should also be noted that starting from a period of 10 ms, the number of calls supported remains constant. This is because the generation of a multiplexed packet is mostly triggered by the arrivals of enough audio chunks to fit into the MTU rather than the expiration of the multiplexing period.

VI. Software Implementation

A software prototype is built to demonstrate the feasibility of our scheme and its compatibility with H.323. Our software runs under the Linux OS. It can act as a gatekeeper, compress RTP headers and multiplex voice data from H.323 clients. We have tested the system with NetMeeting and H.323 Call Generator [21]. It is found that when setting the multiplexing period to 10 ms, the actual delay incurred by our MUX is mostly below 10 ms and is less than 20 ms for some voice packets. This amount of delay is mainly due to the packet waiting at the MUX and the occasional inaccuracy of OS timer. We also observe that the processing times of MUX/DEMUX due to header manipulation and packet transmission are negligible and generally below 1 ms. It is just as what we have assumed previously in the simulations. This prototype shows that our multiplexing system is capable of increasing the bandwidth efficiency with only a small incursion of extra packet delay. We conjecture that an even better performance will result when our system is implemented in hardware.

VII. Conclusion

We have proposed a voice multiplexing system for H.323 IP telephony to improve bandwidth efficiency. Our scheme aggregates voice packets from different sources into a single multiplexed packet before transmission over a bandwidth-limited WAN trunk. The proposed system can effectively utilize the trunk and increase the number of simultaneous calls supported

by several times relative to the conventional scheme. Since the proposed scheme is compliant with the H.323 standard, it can be readily implemented and deployed.

REFERENCES

[1] G. Thomsen and Y. Jani, "Internet telephony: Going like crazy," *IEEE Spectrum*, pp. 52–58, May 2000.

[2] M. Hassan, A. Nayandoro, and M. Atiquzzaman, "Internet telephony: Services, technical challenges, and products," *IEEE Commun. Mag.*, pp. 96–103, Apr. 2000.

[3] Microsoft NetMeeting.. [Online]. Available: http://www.microsoft.com/windows/netmeeting/

[4] K. Sundstrom, A. Rueda, and R. D. McLeod, "Internet telephony compression algorithms," in *Proc. WESCANEX '97*, Winnepeg, MB, pp. 13–18.

[5] R. V. Cox, "Three new speech coders from the ITU cover a range of applications," *IEEE Commun. Mag.*, pp. 40–47, Sept. 1997.

[6] B. Goodman, "Internet telephony and modem delay," *IEEE Network*, pp. 8–16, May/June 1999.

[7] ITU-T Recommendation G.114, "One way transmission time," ITU, Feb. 1996.

[8] P. T. Brady, "Effects of transmission delay on conversational behavior on echo-free telephone circuits," *Bell Labs Tech. J.*, vol. 50, pp. 115–134, 1971.

[9] ETSI GSM 6.10 version 5.1.1, "Digital cellular telecommunications system (phase 2+) full rate speech transcoding," ETSI , May 1998.

[10] ITU-T Recommendation G.723.1, "Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s," ITU, Mar. 1996.

[11] ITU-T Recommendation G.729 Annex A, "C source code and test vectors for implementation verification of the G.729 reduced complexity 8 kbit/s CS-ACELP speech coder," ITU, Nov. 1996.

[12] ITU-T Recommendation H.323 version 4, "Packet-based multimedia communications systems," ITU, Nov. 2000.

[13] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "IETF RFC 3261," SIP: Session initiation protocol, June 2002.

[14] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," in *Proc. IETF RFC 1889*, Jan. 1996.

[15] S. Casner and V. Jacobson, "Compressing IP/UDP/RTP headers for low-speed serial links," in *Proc. IETF RFC 2508*, Feb. 1999.

[16] T. Hoshi, K. Tanigawa, and K. Tsukada, "Proposal of a method of voice stream multiplexing for IP telephony systems," in *Proc. IWS '99*, Feb. 1999, pp. 182–188.

[17] B. Thomspon, T. Koren, and D. Wing, "Tunneling multiplexed compressed RTP ('TCRTP')," *Internet Draft*, Feb. 2002.

[18] R. Pazhyannur, I. Ali, and C. Fox, "PPP multiplexing," IETF RFC 3153, Aug. 2001.

[19] K. Sriram and W. Whitt, "Characterizing superposition arrival processes in packet multiplexers for voice and data," *IEEE J. Select. Areas Commun.*, vol. SAC-4, pp. 833–846, Sept. 1986.

[20] S. Deng, "Traffic characteristics of packet voice," in *Proc. ICC '95*, vol. 3, Seattle, pp. 1369–1374.

[21] H.323 Call Generator. [Online]. Available: http://callgen323.sourceforge.net/

H. P. Sze received the M.Phil. degree from the Department of Information Engineering at the Chinese University of Hong Kong.

From 2000 to 2001, he worked on a research and development project in the area of VoIP applications, at the Chinese University of Hong Kong. He is now working as a Member of the Professional Staff at Hong Kong Applied Science and Technology Research Institute, Kowloon, Hong Kong.

Soung C. Liew (S'84–M'87–SM'92) received the S.B., S.M., E.E., and Ph.D. degrees from the Massachusetts Institute of Technology (MIT), Cambridge.

He is currently a Professor and Director of the Area of Excellence in Information Technology at the Chinese University of Hong Kong. He has had extensive experience in the areas of Internet protocols and applications, multimedia communications, broadband networks, and packet switching. He holds three U.S. patents.

Dr. Liew is a Member of Research Grant Council Engineering Panel of Hong Kong, and Fellow of HKIE. He is listed in Marquis *Who's Who in Science and Engineering*.

Jack Y. B. Lee is with the Department of Information Engineering at the Chinese University of Hong Kong. He directs the Multimedia Communications Laboratory at the Chinese University of Hong Kong (http://www.mcl.ie.cuhk.edu.hk) and conducts research in distributed multimedia systems, fault-tolerant systems, multicast communications, and Internet computing.

Danny C. S. Yip received the B.Eng. (Hon.) and M. Phil. degrees in information engineering from the Chinese University of Hong Kong, in 1998 and 2000, respectively.

He is currently working in a global technology and management consulting firm, Accenture Company Limited, Kowloon, Hong Kong.