AN ASYNCHRONOUS TIME DIVISION MULTIPLEXING SCHEME FOR VOICE OVER IP

By

Yip Chung Sun Danny

A Thesis

submitted in partial fulfillment of the requirements

for the Degree of Master of Philosophy

Division of Information Engineering

The Chinese University of Hong Kong

August 2000

摘要

先進的聲音壓縮技術使語音交通在互聯網上的成本比在傳統的電話網絡的成本更低。當未解決的技術性問題逐一被破解，互聯網上的語音通訊會越見普遍。要全面應用語音通訊於互聯網上，還有一個重要的障礙，這就是使用者所不想經驗到的語音延誤。一個主要延誤的來源是由組成延誤所構成的，而所謂的組成延誤就是集合充足的語音數據來放進一個封包所需要的時間。數據長度越長，組成延誤就會越長。當語音壓縮技術不斷進步，每一位元組就相當於一個更長的語音時間。這樣，若果數據長度不變，組成延誤便會更長。雖然我們可以用較短的數據長度來處理這問題，但是網絡頻寬的運用效率便會很低，因爲每個包封都包含一個爲了傳送而存在的 IP、UDP 及 RTP 包頭。如果每一個語音幀也送出一個封包，包頭的負荷便會令實際的頻寬使用率低至 20%。因此，現時常見的網上電話應用軟件會將幾個語音幀 (約 100 微秒) 放進一個封包內才傳送，而代價便是加長的組成延誤。

在本論文中，我們建議一個新的計劃，名爲非同步分時多工化系統，目標是在保持低組成延誤的情況下改善網絡頻寬效率。在這方案中，一個多工器需要加在源網絡和一個解多工器在目的網器。它們是用來將兩個網絡之間的交通在傳送至遠距離的目的地前多工化。由一些在源網絡的發送者所傳送而目的地在同一目的網絡之語音 RTP 封包會首先被傳送到多工器，多工器會將它們的封包集合成爲一個單一封包。而在原有的語音封包中的 40 位元組長的 RTP/UDP/IP 包頭會被刪除，取而代之的是一個 2 位元組長的微型包頭。一些以上的封包會被集合在一個 IP 封包中，這樣，它們便可以一同分擔 IP 包頭的負荷。因此，源網絡與目的網絡之間的網絡交通便會有效率地運用頻寬。同時，在語音來源的組合延誤亦可保持低的水平。利用這技術，兩個網絡之間便能同時支援更多的連線，且擁有較短的端點延誤。

Acknowledgement

I would like to express my deepest gratitude to my supervisor, Professor Soung-chang Liew, for his constant guidance, insights, invaluable suggestions, constructive criticisms, sustained encouragement and support during the period of my study. His inspiration, networking expertise and nice personality made my Master study a pleasant and fruitful experience.

I would also like to thank all the members of the Broadband Communications Laboratory for their support. Special thanks to Mr. Man-chi Chan and Mr. Tsz-ming Lau for their kind advice on my research work.

In addition, I have to thank my brothers and sisters in Lord. They always give me their whole-hearted support and love.

Finally, I am grateful to my parents and Ms. Mei-kuen Lai for their love and endless support in every aspect of my life.

Abstract

Advanced audio compression techniques have made it possible to carry voice traffic over the Internet at a much lower cost than traditional telephone network. Voice communications over the Internet (voice-over-IP) will become increasingly prevalent, as outstanding technical problems are being solved. One remaining key hurdle to the large deployment of voice-over-IP is the end-to-end delay experienced by the users. A major source of delay can be attributed to the so-called assembly delay: the time required at the source to gather a sufficient amount of voice data to be put into a packet for delivery. The longer the data length, the longer is the assembly delay. As speech compression techniques improve, each byte would correspond to a longer voice duration and the assembly delay would be even worse if the data length were to be kept fixed. Although one could make the data length smaller to overcome this problem, this would result in inefficient use of network bandwidth, since each packet also contains a fixed header, including IP, UDP, and RTP headers for transport purposes. If a packet were sent for every audio frame, the effective bandwidth utilization would be as low as 20% after the header overhead is taken into account. Currently, common Internet telephony applications send a few frames of audio data (around 100 milliseconds) in a single packet at the expense of longer assembly delay.

In this thesis, we propose a novel asynchronous time division multiplexing scheme to improve the network bandwidth efficiency while keeping the assembly delay small in voice-over-IP. In this scheme, a multiplexer and a demultiplexer are added in the source network and the destination network respectively for multiplexing the traffic between the two networks before transport over a long distance. Individual audio RTP packets from a number of sources in the source network destined for a number of destinations in the same destination network are first transmitted to the multiplexer, where they are aggregated into a single packet. The 40-byte RTP/UDP/IP header of each original packet is stripped off at the multiplexer and replaced with a 2-byte mini-header, and a number of such packets are aggregated into an IP packet so that they can shoulder the IP packet-header overhead together. Thus, the network transport between the source and destination networks can be bandwidth-efficient. Meanwhile, the assembly delay at the audio sources can be kept low. With this technique, more simultaneous connections can be supported between two localities with small end-to-end delay.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The development and expansion of the Internet in the last few years is making it possible for real-time voice traffic to reach many different corners of the world over the Internet. As the Internet is not originally designed for real-time communications, real-time voice traffic over the Internet faces a number of technical challenges. These are due to the lack of Quality of Service (QoS) guarantee in the Internet in terms of bandwidth, packet loss rate, delay and jitter. In general, QoS guarantee is difficult over the legacy Internet [1]. With the development of QoS implementation in the Internet such as MultiProtocol Label Switching (MPLS) [2], Differentiated Services (DiffServ) [3], and Resource ReSerVation Protocol (RSVP) [4], the hindrance to real-time voice over IP can be overcome in the near future.

Delivering voice over IP, first of all, has the advantage of cost reduction for long distance telephone call because it can bypass the high international toll charge from telecommunication companies and use the close-to-free Internet. Moreover, voice over IP can be enhanced and integrated with other applications such as video, white boarding, calendar tools within the same data network. When IP telephony starts to be popularly used, real-time voice traffic will not be negligible in the Internet. The bandwidth efficiency of voice communication should be optimized for better network utilization.

There has been growing interest in using Real-time Transport Protocol (RTP) to establish real-time network communications over the Internet since it was published by the Internet Engineering Task Force (IETF) as an RFC [5]. In particular, interactive audio communication applications such as Internet phone are adopting this protocol as their underlying protocol. However, there is a concern that the 12-byte RTP header together with 28-byte IP-UDP header is a large overhead for small-sized audio payloads. For example, the ITU G.723.1 codec [6], which is a common speech compression standard for voice communication applications, generates frames of size 20 bytes in every 30 ms interval. Thus, if every frame is sent in an RTP packet, the bandwidth efficiency will be as low as 33%. It is undesirable to the data network when voice traffic becomes prevalent in the future.

Conventional audio communication applications tackle this problem by embedding several audio frames into a single packet in order to increase the ratio of data to header and in other words, increase the bandwidth efficiency. A drawback of this solution is that the assembly time of a packet will be longer because it is directly proportional to the number of audio frames to be embedded in the packet. As audio frame is generated after raw audio signals in a frame period are captured and encoded, embedding one more audio frame means adding one frame period in the assembly delay of this packet. This delay contributes to the overall end-to-end delay of audio delivery. Recommendation G.114 of the International Telecommunication Union – Telecommunication Standardization Sector (ITU-T) [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 given as 150 ms. A voice-over-IP system with long end-to-end delays causes users to engage in double talk and mutual silence more often [8]. Therefore, the magnitude of the increase in assembly delay is significantly undesirable in interactive voice communications. Thus, we have a dilemma in which either we have large header overhead or large delay, which is also claimed in [9]. Figure 1.1 shows this tradeoff in the G.723.1 audio codec.

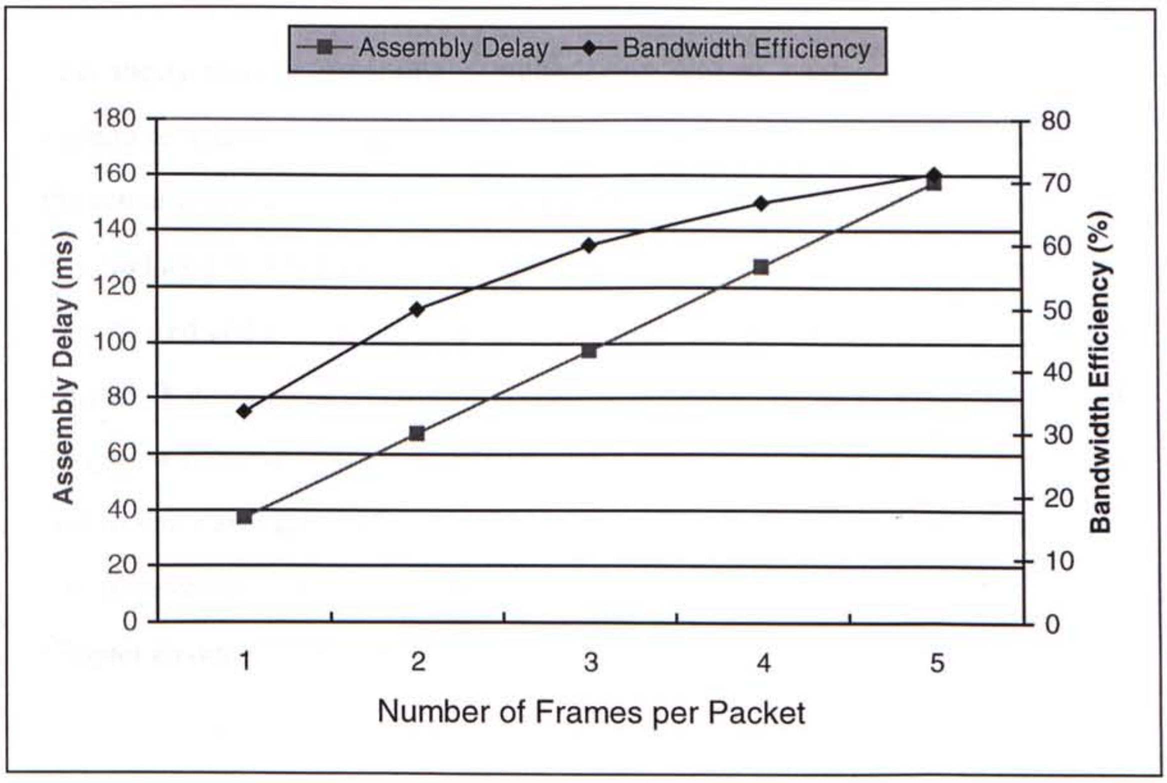


Figure 1.1: Tradeoff between bandwidth efficiency and delay in conventional scheme.

To circumvent the above tradeoff, we propose in this thesis a novel scheme called "Asynchronous Time Division Multiplexing Scheme" (ATDM), which makes it possible to achieve low delay and high bandwidth efficiency at the same time. In this way, real-time conversation can be supported by the data network without large header overhead.

1.2 Organization of Thesis

This thesis aims at proposing a multiplexing scheme applying to voice-over-IP systems to achieve high bandwidth efficiency without increasing delay much. Before the actual discussion of the proposed ATDM scheme, some background knowledge is introduced in Chapter 2. Some basic properties of common speech codecs are summarized and an existing RTP/UDP/IP header compression scheme is presented. Chapter 3 gives some criteria of scenarios in which our scheme can be applied. Chapter 4 describes our proposed scheme in great detail. Chapter 5 focuses on the performance evaluation of our ATDM scheme with the use of simulations and states the improvement of bandwidth utilization over conventional methods. Lastly, Chapter 6 concludes this thesis and presents a possible extension to this scheme.

Chapter 2

Background

2.1 Speech Codec

The primary function of an audio codec is to convert analog voice signal to digital data (coding) and vice versa (decoding). In addition, audio codecs perform voice compression to reduce the bandwidth requirement for supporting voice transmission over digital data networks.

Before discussing the scheme, we have to investigate the characteristics of the source audio data stream so as to customize the scheme to fit the interactive audio communication applications.

To date, there are three speech codecs, which are commonly used in audio communication applications due to their low bit rate nature. They are GSM 6.10 [10], ITU G.723.1 [6] and G.729A [11].

Table 2.1 summarizes the attributes of these three audio codecs: [12]

Codec	GSM 6.10	G.723.1	G.729A
Bit rate (kb/s)	13	5.3 / 6.4	8
Frame size (bytes)	32.5	20	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	Fairly Good	Fairly Good	Good

Table 2.1: The attributes of three common audio codecs.

We can see that different audio codecs have different bit rates, frame periods, processing delays and audio qualities. However, they have a common characteristic. They all generate constant bit rate audio data frames with constant frame period. Even when a silence suppression scheme is employed, the codecs still generate 2-state variable bit rate audio data frames with constant frame period – a lower constant bit rate for silent state and a higher constant bit rate in active state. We will utilize this property in designing our multiplexing scheme.

2.2 RTP/UDP/IP Header Compression

2.2.1 Real-Time Transport Protocol

Real-Time Transport Protocol (RTP) which is defined in IETF RFC [5], provides end-to-end network transport functions suitable for applications transmitting real-time data, such as interactive audio and video, usually over User Data Protocol (UDP). RTP provides payload type identification and most importantly, timestamping and sequence numbering which are important in synchronization and re-sequencing on the receiver side before actual playback.

Most Internet phone applications used RTP/UDP/IP as the underlying protocol for the transmission of audio stream data packets. Every audio frame generated from the encoder on the sender side will be encapsulated by an RTP header with sequence number and encoding timestamp. This RTP packet will be sent to the receiver using UDP. Once the receiver gets this packet, it uses the sequence number for re-sequencing the packets in playback buffer because of the out-of-order effect and synchronizes the playback time with other stream sources like video with the timing information from timestamp stored in the RTP header.

2.2.2 RTP/UDP/IP Header Compression

The total header length of an RTP/UDP/IP packet is 40 bytes, which is a large overhead for small audio data payloads such as 20 bytes for G.723.1 audio frame. Thus, Casner and Jacobson proposed an RTP/UDP/IP header compression scheme to increase the bandwidth utilization and minimize the end-to-end delay [13]. This can reduce the RTP/UDP/IP header size from 40 bytes down to 2 bytes and it compresses RTP/UDP/IP headers on a link-by-link basis. A limitation is that it must be supported by the link layer hardware to interpret the special format of the payload on the link layer.

The algorithm used in the RTP/UDP/IP header compression is based on two important observations on the packet header of an RTP stream.

The first is that many essential fields in the IP, UDP and RTP headers do not change over the lifetime of an RTP connection. They include IP source and destination addresses, UDP source and destination port numbers, and the RTP SSRC field. We denote the combination of these five fields as the session context. After sending the uncompressed header once, these fields can be eliminated from the subsequent compressed headers and is represented by a session context number instead.

The second observation is that the remaining RTP fields like sequence number and timestamp will change in a predictable manner for real-time audio stream. The difference of these fields between consecutive packets is often constant and therefore the second-order difference is zero. Hence, differential coding can be applied on these fields. In other words, by maintaining the uncompressed header and the first-order differences in the session state synchronized between the compressor and the decompressor, the decompressor can reconstruct the original fields simply by adding the first-order differences to the stored uncompressed fields as each compressed packet is received.

According to the above observations, only a session context number and a link sequence number are sent in a compressed header of 2 bytes length instead of the original 40-byte long RTP/UDP/IP header.

In our multiplexing scheme, we will adopt a similar header compression scheme to minimize the header length of all embedded audio packets of different streams inside a multiplexed packet.

Chapter 3

Scenario and Assumptions

Our scheme is based on two observations:

1. The LAN bandwidth is usually quite abundant and efficient usage of the bandwidth is usually not a major concern.
2. When voice over IP becomes prevalent, there will be many simultaneous voice sessions between two localities separated by a long distance.

With these observations, we investigate techniques that can be applied on the voice-over-IP traffic to improve the WAN network utilization. Compression and multiplexing are important in achieving this goal. While compression algorithms focus in representing the user's data with minimum number of bits, multiplexing techniques attempt to keep the transmission protocol overhead to a minimum.

Compression schemes for speech have been well developed, achieving high compression ratio and good quality of speech. Many speech codecs have already been standardized by the International Telecommunication Union (ITU). As speech compression techniques improve, each byte can represent a longer voice duration and the assembly time of a packet will get longer for a fixed data length. Although one can make the data length smaller to overcome the problem of long packet assembly delay, this will result in inefficient use of network bandwidth, since each RTP packet also contains a fixed-sized header.

Multiplexing is another key mechanism to reduce the header overhead so as to utilize the WAN network better. Multiplexing approach is based on the assumptions that there are multiple sessions between two localities and that the payloads of the data packets are relatively small compared to the overhead imposed by the network to transport data between the sender and the receiver. Voice traffic over the Internet often gives rise to the above two situations.

As such, we can use a large-overhead approach within a LAN. In other words, every audio frame is sent in an RTP packet from senders. However, voice packets bound for a common remote area will be multiplexed into a packet to reduce header overhead. Thus, a packet going through a long-distance network will contain data from several voice sources and they can co-shoulder the IP header overhead. The main idea of this scheme is illustrated in Figure 3.1, in which the bandwidth at the WAN link is used much more efficiently while end-to-end delay of each connection does not increase much because we do not need to wait for several frames before packetization as adopted in the conventional Internet phone applications.

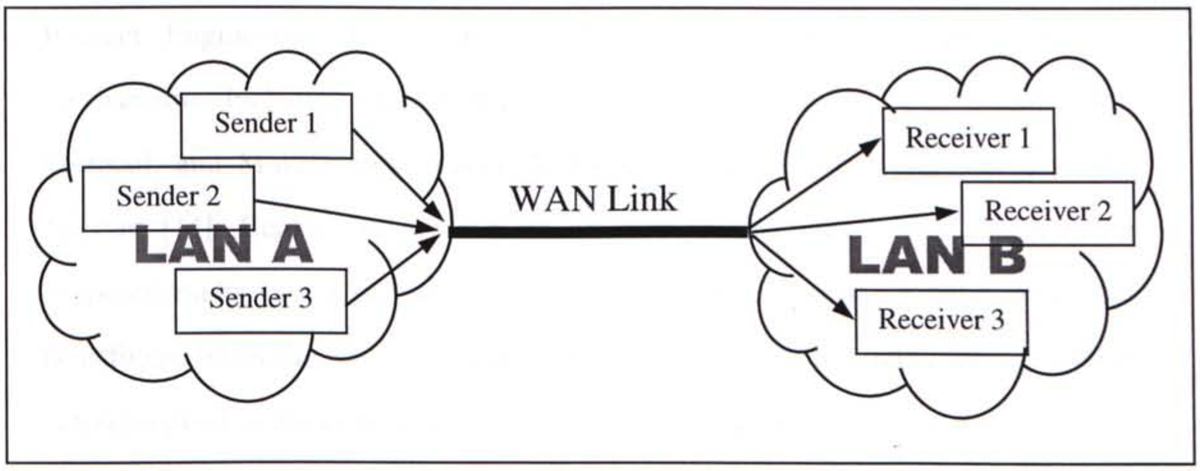


Figure 3.1: An overview of network environment.

The bandwidth of the WAN link is often an expensive resource and constitutes the bottleneck of communications between two sites. Therefore, this scheme is beneficial to many network situations because it can achieve high bandwidth utilization over the WAN link.

This multiplexing scheme performs best when the network jitter of the WAN link is small. Otherwise, the large network jitter may affect the demultiplexer in recovering the RTP packet on the receiver side network. Therefore, the WAN link should provide this kind of QoS guarantee. Network jitters occur because of the variability of queuing delays in the network and the propagation delay differences when packets of the same stream traverse over different paths in the Internet. These two circumstances can be avoided for corporate WAN IP networks since the queuing delay can be reduced by using a faster link and the routing table can be defined within the corporate. However, in the Internet, some mechanisms must be incorporated to control the traffic in order to provide services with jitter bound. The Internet Engineering Task Force (IETF) is working on a number of such mechanisms, including Differentiated Services (DiffServ), Resource ReSerVation Protocol, and MultiProtocol Label Switching (MPLS), to provide QoS over the Internet [14]. Before the deployment of such architectures in the Internet, the proposed multiplexing scheme can perform well in the corporate network. In the near future when the Internet can provide jitter guarantee, the multiplexing scheme can be applied to the Internet for better network utilization.

Chapter 4

Asynchronous Time Division Multiplexing Scheme

4.1 Basic Idea

In this scheme, a multiplexer and a demultiplexer are added to the sender LAN and the receiver LAN respectively. All out-bound voice packets are sent from the senders to the multiplexer instead of directly to the receivers. The multiplexer aggregates voice data from different sources into a single packet. This multiplexed packet is then sent to the demultiplexer on the receiver LAN through the WAN link. Once the demultiplexer receives the multiplexed packet, the data will be restored to individual voice packets with the original RTP header and destination information, and these packets will then be sent to their respective receivers.

The scheme is illustrated in Figure 4.1:

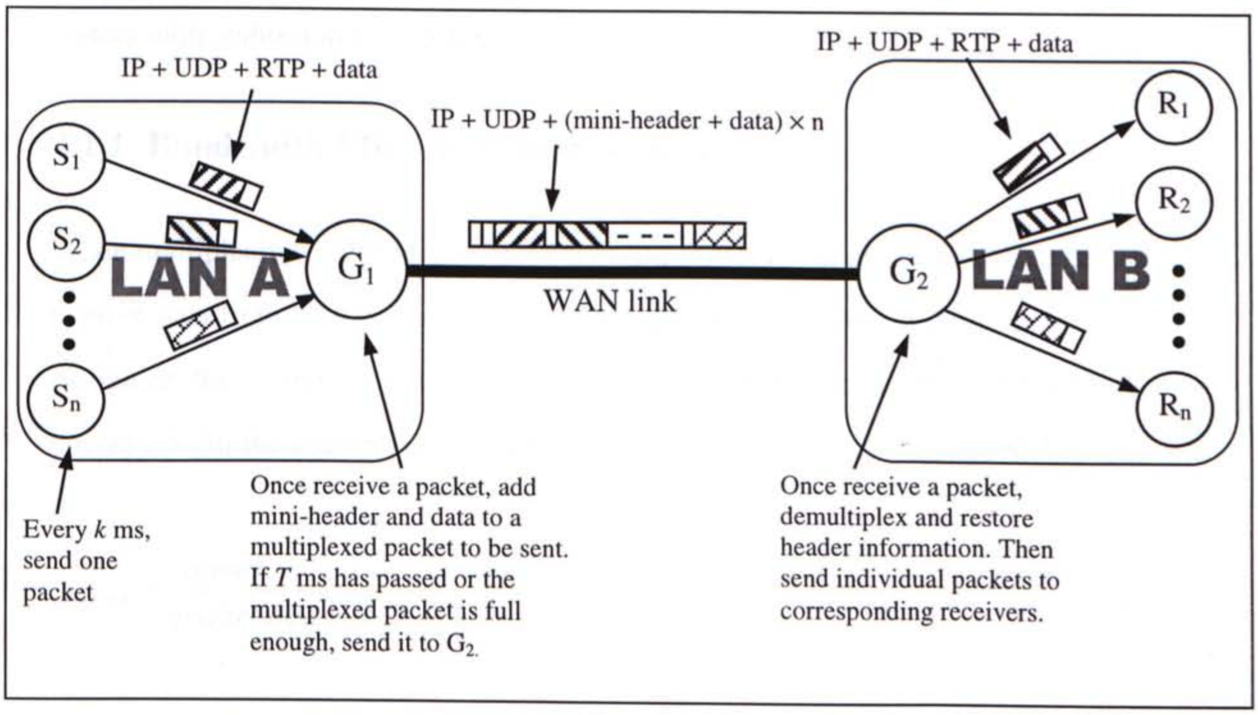

IP + UDP + RTP + data
IP + UDP + (mini-header + data) × n
IP + UDP + RTP + data
S_1
S_2
S_n
LAN A
G_1
WAN link
G_2
LAN B
R_1
R_2
R_n
Every k ms, send one packet
Once receive a packet, add mini-header and data to a multiplexed packet to be sent. If T ms has passed or the multiplexed packet is full enough, send it to G_2.
Once receive a packet, demultiplex and restore header information. Then send individual packets to corresponding receivers.

Figure 4.1: The mechanism of ATDM scheme.

Basically, the multiplexer multiplexes the audio RTP packets and sends the multiplexed packet out every T ms which is set to the audio frame period from senders (k ms). However, the multiplexer may send out a multiplexed packet before T ms expires if many audio packets arrive within a short time and the multiplexed packet has reached the link layer maximum frame size of the WAN link. This is to avoid increase in probability of packet loss due to the fragmentation of the multiplexed packets in WAN link. One point that should be noted is that it is not an absolute must to set the multiplexing period, T ms, to be the same as the audio frame period, although it is convenient to do so when the codecs are of the same type with the same frame period. By making T flexible, the multiplexing system can also be applied to situations in which the senders are not homogeneous and use speech codecs with audio frames of different durations.

4.1.1 Bandwidth Efficiency Improvement

When multiplexing voice packets from different sources, a mini-header is inserted in front of data from each source. This mini-header is essential for the demultiplexer to recognize the source of the data and restore the important RTP, UDP, and IP headers. With the multiplexing scheme, the bandwidth efficiency of the WAN link is:

$$\rho_{ATDM} = \frac{data\ size}{packet\ size} = \frac{a \times n}{h_{UDP} + (h_m + a) \times n} = \frac{a}{\left(\frac{h_{UDP}}{n} + h_m\right) + a} \qquad \text{— (4.1)}$$

where a = audio data size

h_m = mini-header size

h_{UDP} = UDP + IP header size of the packet = 28 bytes

n = number of sources multiplexed

Because h_{UDP} cannot be changed, the size of a mini-header plays a key role in the bandwidth efficiency in the WAN link. Our scheme adopts RTP/UDP/IP header compression to reduce the mini-header size down to 2 bytes. The details of RTP/UDP/IP header compression in constructing mini-header will be discussed in the later section.

Conventional methods send packets from source to destination without multiplexing. If one RTP packet is sent for every audio frame, the bandwidth efficiency is:

$$\rho_{old} = \frac{data\ size}{packet\ size} = \frac{a}{h_{RTP} + a} \qquad \text{— (4.2)}$$

where a = audio data size

h_{RTP} = RTP + UDP + IP header size of the packet = 40 bytes

It can be seen that the bandwidth efficiency of the WAN link can be increased by using the multiplexing scheme with header compression. In fact, the bandwidth efficiency can be improved quite significantly when the number of concurrent connections is large.

Take audio compression standard G.729A for example, each audio frame is 10 bytes and corresponds to 10 ms of speech. Assume that each packet contains one audio frame, the data size of each packet is 10 bytes. If there are 50 simultaneous active connections between the two localities, the bandwidth efficiency of the WAN link with ATDM scheme is:

$$\rho_{ATDM} = \frac{10}{\left(\frac{28}{50} + 2\right) + 10} = 0.796$$

In contrast, the bandwidth efficiency without the ATDM scheme is:

$$\rho_{old} = \frac{10}{40 + 10} = 0.2$$

The bandwidth efficiency is increased by about 300%.

Since the bandwidth efficiency in a conventional scheme is very low (0.2 in the typical example above) if one audio frame is put in each RTP packet. Therefore, to keep overhead small, common Internet phone applications embed several audio frames into each packet to ensure a large a value in Eq. (4.2). The tradeoff, however, is that this increases the assembly time of a packet and the overall end-to-end delay is in turn increased. The delay issue will be discussed in the next section.

4.1.2 Delay Reduction

With ATDM, senders can put a small audio frame into a single RTP packet without worrying the bandwidth efficiency of the WAN link. This minimizes the packet-assembly time, which can be a significant component in the overall end-to-end delay [15].

On the other hand, the multiplexer will introduce a waiting delay time, which corresponds to the time needed to wait for the arrival of a group of packets to be multiplexed. In our scheme, we limit the delay to one audio-frame time, k ms. Because of silence periods, some sources may not generate any packets during a k ms interval. The multiplexer will multiplex the arriving packets from different senders at the end of each k ms. Since the processing time of multiplexing and demultiplexing is insignificant, the delay introduced by adding the multiplexer and the demultiplexer will be bound by k ms i.e. one audio frame time if the WAN link is not overflowed. However, the delay can go beyond this bound if the WAN link is congested and the multiplexed packets may need to queue up to be transmitted over the link. If the number of simultaneous active connections is larger than that can be supported by the WAN link, the system becomes unstable.

Our simulation investigations will be presented in detail in Chapter 5. Our simulation generates a set of results presented as a graph of the overall delay against WAN link utilization, where it is shown that the delay performance of ATDM is comparable to the conventional one-audio-frame-per-packet scheme but the bandwidth efficiency can be improved by several hundred percent.

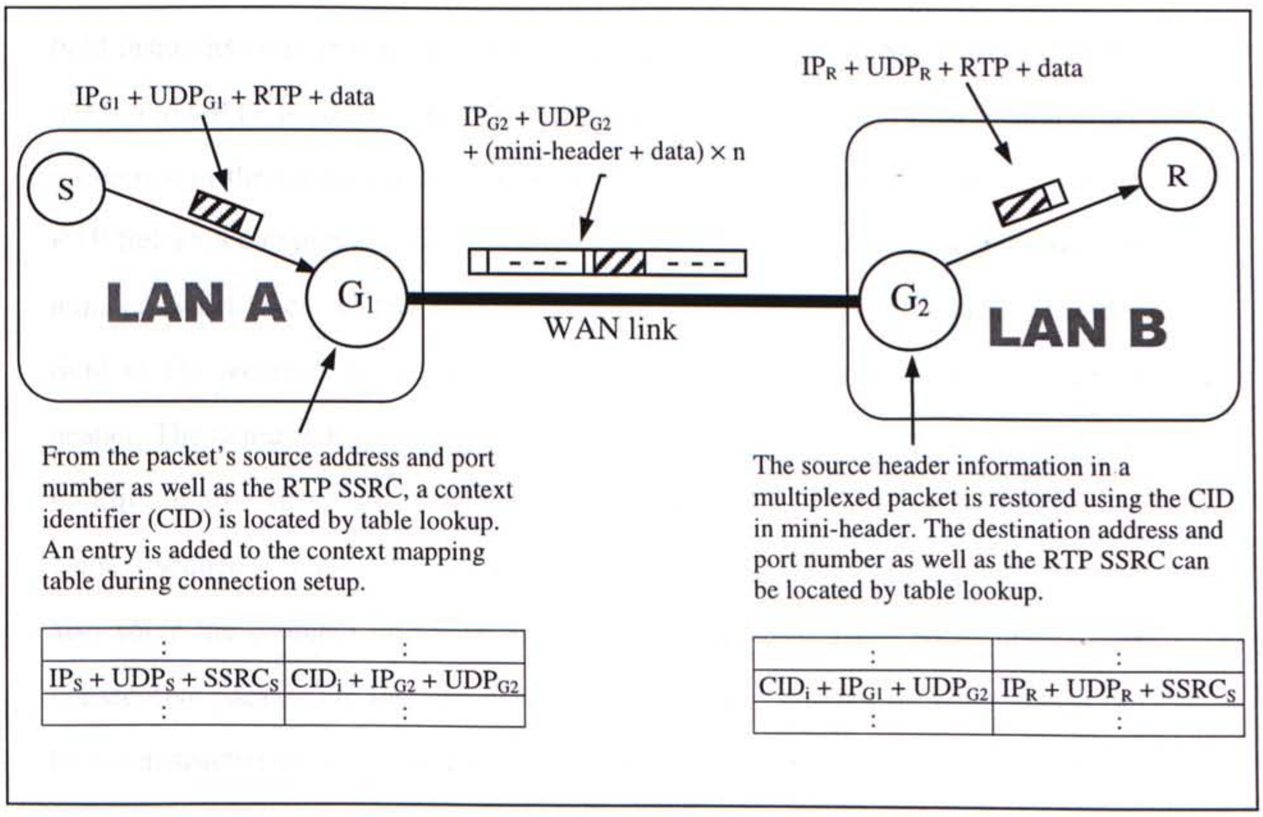
4.2 Header Compression

In ATDM, header compression is a key to improving the bandwidth efficiency of the WAN link. Header compression algorithm used in this scheme draws heavily upon the design of RTP/UDP/IP header compression as described in [13], which has been introduced in Section 2.2.2.

This algorithm is based on two important observations on RTP, UDP and IP headers. The first is that many essential fields in the IP, UDP and RTP headers do not change over the lifetime of an RTP connection. The second observation is that the remaining RTP fields like sequence number and timestamp will increase linearly over the lifetime of an audio stream.

In the design of RTP/UDP/IP header compression in RFC2508, *link-layer* compression is targeted. However, many of the ideas are also applicable to our application-layer considerations. As such, we adapt RFC 2508 and complement it with new ideas to arrive at a compression protocol suitable for our application.

In our proposed scheme, many fields in the UDP and IP headers can be changed without affecting the receiver playback of audio packets. In the UDP and IP headers, the two important elements to be reconstructed are destination address and port number for routing purpose. The fields in RTP header must all be reconstructed by our demultiplexer without any loss like that implemented in RFC2508; otherwise the timing information of the audio payload will get lost. Figure 4.2 illustrates how context identifier in the mini-header is mapped to the addresses and port numbers of sender and receiver as well as the RTP SSRC in multiplexer and demultiplexer.


$IP_{G1} + UDP_{G1} + RTP + data$
$IP_{G2} + UDP_{G2}$ + (mini-header + data) × n
$IP_R + UDP_R + RTP + data$
S
R
LAN A
G_1
WAN link
G_2
LAN B

From the packet's source address and port number as well as the RTP SSRC, a context identifier (CID) is located by table lookup. An entry is added to the context mapping table during connection setup.

:	:
$IP_S + UDP_S + SSRC_S$	$CID_i + IP_{G2} + UDP_{G2}$
:	:

The source header information in a multiplexed packet is restored using the CID in mini-header. The destination address and port number as well as the RTP SSRC can be located by table lookup.

:	:
$CID_i + IP_{G1} + UDP_{G2}$	$IP_R + UDP_R + SSRC_S$
:	:

Figure 4.2: An overview of context identifier matching in multiplexer and demultiplexer.

4.2.1 Header Compression Process

In our scheme, a compressed mini-header with 2-byte size consists of a session context identifier (CID) and a context sequence number. The process of header compression and decompression is very simple and fast to implement. When the multiplexer receives an audio RTP packet from a sender, it will search for a matched session context by the source IP address and port number as well as the RTP SSRC field inside its context mapping table, which will be defined in Section 4.2.2. If the fields in the RTP header can fulfil the criteria of compression (i.e. zero second-order difference of the timestamp field in the RTP header and no changes in the constant RTP fields), a compressed mini-header can be set with a CID and a context sequence number, which is calculated from the difference between the RTP sequence number field in the received packet and that previously sent in a synchronization mini-header. The demultiplexer can reconstruct all the RTP header information by adding multiple of the first-order differences to the saved RTP header according to the context sequence number inside the mini-header. The context sequence number can also solve the problem of packet loss in a limited sense. Even when one or few consecutive packets are lost, the correct RTP sequence number of later packets can be reconstructed by using the context sequence number; the correct RTP timestamp can also be recalculated by adding multiple of first-order difference to the previously stored timestamp according to the sequence number difference. We will give an example in regenerating the RTP sequence number and timestamp with the use of context sequence number in Section 4.2.2. By applying differential coding on the RTP timestamp field, we can keep the size of a compressed mini-header 2 bytes only so that we can improve the bandwidth efficiency of the link between the multiplexer and the demultiplexer under our ATDM scheme.

When the second-order difference of the RTP timestamp field is not zero, a synchronization mini-header will be placed in front of the audio payload in the multiplexed packet instead of a compressed mini-header. This synchronization mini-header includes the sequence number and timestamp fields inside the RTP header besides the session context identifier and the context sequence number. As such, the demultiplexer can use the additional RTP sequence number and timestamp to recover the RTP header in the audio packet to the receiver. This should rarely happen in the voice traffic when using common audio codecs because they all generate audio data frames with constant frame period.

At the beginning of a voice-over-IP session, an uncompressed mini-header has to be put in front of the audio data inside the multiplexed packet. This uncompressed mini-header basically includes a full RTP header from the received packet, a session context identifier and a context sequence number. This uncompressed mini-header will also be sent occasionally if there is any change in any of the RTP fields that are expected to be homogeneous throughout the session in the headers of audio packets from the sender. However, in the lifetime of common voice-over-IP applications, this would rarely happen.

In the RTP header, there is a one-bit field called marker or M which will be set on the first packet of each audio talkspurt. This one bit will be carried in the compressed mini-header because if it were treated as a constant field such that each change would require the sending of an uncompressed mini-header, compression gain will be substantially reduced.

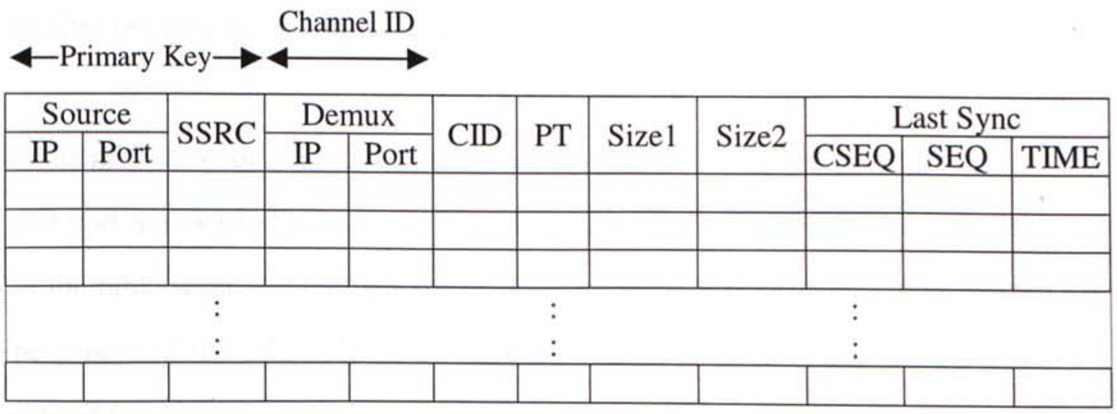
When using differential coding, an error will be propagated and accumulated. To tackle the accumulation of error by differential coding, a synchronization mini-header with audio payload as described before will be sent periodically inside a multiplexed packet from the multiplexer. Period of sending synchronization packet is the same as the context sequence number cycle. Although this will reduce the compression gain, this periodic synchronization is necessary in tackling the error propagation problem induced by differential coding.

4.2.2 Context Mapping Table

Figure 4.3 and Figure 4.4 show the context mapping tables used in the multiplexer and demultiplexer respectively. All the abbreviations used in the two mapping tables are listed in Table 4.1. Context mapping tables store the important information needed in multiplexing and demultiplexing.

Abbreviation	**Description**
Source IP	Sender IP Address
Source Port	Sender Port Number for Sending Audio RTP Packets
SSRC	Synchronization Source Identifier in the Sender RTP Stream
Demux IP	Demultiplexer IP Address
Demux Port	Demultiplexer Port Number
CID	Context Identifier
PT	Payload Type in the Sender RTP Stream
Size1	Payload Size in Active State
Size2	Payload Size in Idle State
Last Sync CSEQ	Context Sequence Number in Last Synchronization Mini-header
Last Sync SEQ	RTP Sequence Number in Last Synchronization Mini-header
Last Sync TIME	RTP Timestamp in Last Synchronization Mini-header
Mux IP	Multiplexer IP Address
Dest IP	Destination Receiver IP Address
Dest Port	Destination Receiver Port Number for receiving RTP Packets
Time Diff	First-Order Difference of RTP Timestamp
RTP Header	Whole RTP Header in Last Uncompressed Mini-header

Table 4.1: Abbreviations used in the context mapping tables.


Channel ID
Primary Key

Source		SSRC	Demux		CID	PT	Size1	Size2	Last Sync		
IP	Port		IP	Port					CSEQ	SEQ	TIME
		⋮				⋮			⋮		

Figure 4.3: Context mapping table used in the multiplexer.

Channel ID
←→
←Primary Key→

Mux IP	Demux Port	CID	Dest		Size1	Size2	Time Diff	RTP Header	Last Sync		
			IP	Port					CSEQ	SEQ	TIME
		:				:			:		

Figure 4.4: Context mapping table used in the demultiplexer.

A record is added to both mapping tables in multiplexer and demultiplexer during the connection establishment stage. The details of connection establishment are described in section 4.4. The fields like Last Sync and RTP Header will be filled in after sending the uncompressed packet in the beginning.

In the multiplexer, the primary key used for identifying a context session is the combination of the RTP synchronization source identifier, the source IP address and port number in the audio RTP packets from the senders. After a match is found in the table when the multiplexer receives an audio RTP packet, a mini-header can be generated with the help of the stored information in the table. By matching the size of the received RTP payload with Size1 and Size2, the multiplexer can set the state bit in the mini-header. By calculating the difference between the Last Sync SEQ and the RTP sequence number in the received packet, the multiplexer can compute the context sequence number for this packet. Then, it can multiplex this packet to the target ATDM channel identified by the demultiplexer IP address and port number. If the multiplexer sends out a synchronization mini-header or an uncompressed mini-header, it will set the Last Sync values accordingly.

In demultiplexer, the primary key used for identifying a context session is the combination of the multiplexer IP address, the demultiplexer receiving port number and the context identifier stored in every mini-header. An ATDM channel can be identified by the combination of multiplexer IP address and the demultiplexer receiving port number. In other words, one multiplexer can establish multiple ATDM channels by sending to different port number of the same demultiplexer. After a match is found when the demultiplexer receives a multiplexed packet, it can regenerate the embedded packets one by one and send them to the corresponding receiver clients. With the state bit in the mini-header, the demultiplexer can know the audio payload size after the mini-header. By using the Last Sync values and the context sequence number sent in a compressed mini-header, the RTP sequence number and timestamp fields can be recovered. Together with the constant RTP fields in the RTP header stored in the mapping table, the whole RTP header can be regenerated. Then, the demultiplexer can send this RTP packet to the corresponding receiver according to the destination IP address and port number. When the mini-header is of synchronization or uncompressed type, the Last Sync values will be updated accordingly. If the mini-header is of uncompressed type, the RTP Header stored in the context mapping table will be updated by the received RTP header after the mini-header.

An example of context sequence number calculation in multiplexer and RTP sequence number and timestamp regeneration in demultiplexer is given below for better illustration.

At a particular time, the context mapping tables of multiplexer M and demultiplexer D are shown in Figure 4.5.

Source		SSRC	Demux		CID	PT	Size1	Size2	Last Sync		
IP	Port		IP	Port					CSEQ	SEQ	TIME
IP_A	$Port_A$	X	IP_D	7000	10	P	10	4	2	40	14200
		:				:			:		
		:				:			:		

(a) An example of context mapping table in multiplexer M

Mux IP	Demux Port	CID	Dest		Size1	Size2	Time Diff	RTP Header	Last Sync		
			IP	Port					CSEQ	SEQ	TIME
IP_M	7000	10	IP_B	$Port_B$	10	4	10	...	2	40	14200
		:				:			:		
		:				:			:		

(b) An example of context mapping table in demultiplexer D

Figure 4.5: An example of context mapping tables

When an audio RTP packet with sequence number equal to 43 and timestamp equal to 14230 is received in the multiplexer, it will try to calculate the context sequence number for the compressed mini-header to be embedded in the multiplexed packet.

Context sequence number = CSEQ = 43 – Last Sync SEQ + Last Sync CSEQ

= 43 – 40 + 2

= 5

Multiplexer M multiplexes the audio payload preceded by its mini-header with CID equal to 10 and CSEQ equal to 5. It then sends the multiplexed packet to the 7000 port of demultiplexer D.

When demultiplexer D receives this multiplexed packet, a record can be found matched. It tries to regenerate the RTP packet by calculating the changing fields, i.e. sequence number and timestamp.

Sequence number = SEQ = CSEQ – Last Sync CSEQ + Last Sync SEQ

= 5 – 2 + 40

= 43

Timestamp = TIME = (CSEQ – Last Sync CSEQ) × Time Diff + Last Sync TIME

= (5 – 2) × 10 + 14200

= 14230

Therefore, it can be seen that the sequence number and timestamp fields in the RTP header can be successfully recovered in the demultiplexer by using the context sequence number in the mini-header and the context mapping tables.

4.3 Protocol

The protocol defined here is on top of the Internet UDP/IP layer, which takes care of routing. It is used for communications between the multiplexer and the demultiplexer. The target of this protocol is to maintain a set of shared fields in a consistent state between the multiplexer and demultiplexer and to restore the important fields inside the RTP/UDP/IP header of an audio packet. A separate session context is assigned for each RTP/UDP/IP packet stream, as defined by a combination of the IP source address, UDP source port number, and the RTP SSRC field. Each context is identified by an 8-bit identifier, so the maximum number of concurrent streams supported is 256. We extend this supported number of streams by using different UDP port number at the demultiplexer as different ATDM channels. Therefore, each channel can support 256 concurrent streams and one demultiplexer can support multiple channels.

Number of supported stream connections in one channel should be chosen appropriately because the larger the number of connections in one channel that can be supported, the more the multiplexing gain will be, but at the same time the more the size of a mini-header will be, which in turn lowers the bandwidth efficiency of an ATDM channel. When a channel between a multiplexer and a demultiplexer is fully occupied with 256 connections, the effective aggregate average bit rate will be 819 kbps if the bit rate of each audio stream is from ITU G.729A codec of 8 kbps bit rate and with activity rate of 40%. When IP/UDP headers and mini-headers are counted in, the total aggregate average bit rate will be about 1 Mbps. This is already enough for most environments. As multiple channels can be built between a multiplexer and a demultiplexer by using different port, this system can still be deployed even in gigabit WAN networks.

Both uncompressed and compressed mini-header must carry a context identifier and a 4-bit context sequence number used to reconstruct the changing RTP fields. Each context is stored in the context mapping tables on both multiplexer and demultiplexer and is synchronized periodically. The details of context mapping tables have been discussed in Section 4.2.2. This context mapping table is important in mapping the context identifier and constructing the context sequence number in the mini-header in the multiplexer. By using the context identifier and the context sequence number in the mini-header, the demultiplexer can regenerate the RTP packet according to the synchronized context mapping table.

Also, every mini-header must include a state bit for indicating the audio payload size with the help of the context mapping table. Because a speech codec is a 2-state variable-bit-rate source when silence suppression is activated, the size of an audio frame can be of 2 different sizes – larger size when speaking and smaller size when idle. When this state bit is zero, it indicates that the following audio payload will be of smaller size, i.e. an idle-state audio frame. When this state bit is one, it shows that the audio payload will be of larger size, i.e. a talkspurt-state audio frame.

Three different types of mini-header formats are defined for communication between the multiplexer and the demultiplexer. They are identified by the first two bits of a header. They are UNCOMPRESSED_RTP, SYNCHRONIZATION, and COMPRESSED_RTP mini-headers.

4.3.1 UNCOMPRESSED_RTP Mini-Header

UNCOMPRESSED_RTP contains the full RTP header information together with the state bit (S), the context identifier (CID) and the context sequence number (CSEQ). This will be sent by the multiplexer at the start of every session context. This will also be sent when any of the fields in RTP which is expected to be constant changes.

When the multiplexer needs to send a UNCOMPRESSED_RTP mini-header for a particular context, this UNCOMPRESSED_RTP mini-header together with its audio data will be sent in a multiplexed packet from the multiplexer.

Figure 4.6 shows the format of the UNCOMPRESSED_RTP mini-header.

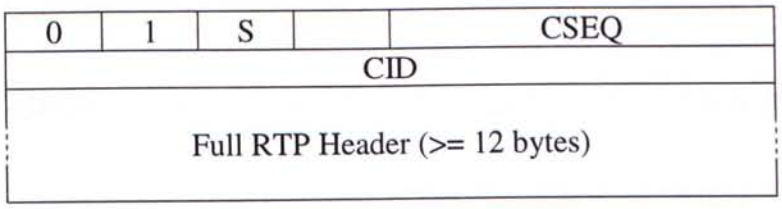


Figure 4.6: Format of the UNCOMPRESSED_RTP mini-header.

4.3.2 SYNCHRONIZATION Mini-header

SYNCHRONIZATION consists of the RTP marker flag (M), sequence number (RSEQ) and timestamp (RTIME) with the state bit (S), the context identifier (CID) and sequence number (CSEQ). This will be sent by the multiplexer periodically. Period of sending synchronization packet is the same as the context sequence number cycle (i.e. once every 16 packets). This is for periodical synchronization of the important fields in RTP such that the problem of error propagation from the use of differential coding applying on them can be alleviated.

The format of the SYNCHRONIZATION mini-header is shown in Figure 4.7.

1	0	S	M	CSEQ
CID				
RSEQ (2 bytes)				
RTIME (4 bytes)				

Figure 4.7: Format of the SYNCHRONIZATION mini-header.

4.3.3 COMPRESSED_RTP Mini-header

COMPRESSED_RTP is the full compression of RTP header and is sent by the multiplexer most of the time. The COMPRESSED_RTP mini-header includes only the state bit (S), the context identifier (CID) and sequence number (CSEQ) and the M flag in RTP. This will be sent with the audio payload when the second-order difference of the RTP timestamp is zero, which normally applies to voice RTP packet stream with constant frame period.

Figure 4.8 shows the format of the COMPRESSED_RTP mini-header.

1	1	S	M	CSEQ
CID				

Figure 4.8: Format of the COMPRESS_RTP mini-header.

4.4 Connection Establishment

In ATDM, connection setup is required before packets from a session can be multiplexed. During connection setup, a session context is allocated and a context identifier (CID) is assigned and recorded at the multiplexer and demultiplexer. The connection setup is required before the client programs can send audio RTP packets out. After the connection is set up, the sender can send audio RTP packets to the multiplexer and with the use of multiplexer and demultiplexer, the receiver can get the corresponding audio RTP packets.

The connection setup is divided into two phases: the addressing phase and the connection phase. If both the addressing and connection phases are successful, the sender can start sending audio RTP packets with the use of multiplexer and demultiplexer. However, if either of the addressing phase or the connection phase fails, the sender can still send audio RTP packets directly destined to the receiver side.

Our connection establishment is incorporated into the current Session Initiation Protocol (SIP), a signaling protocol that creates, modifies, and terminates phone call sessions over the Internet [16], [17]. SIP is a simple, flexible, extensible and powerful protocol that could provide a framework for complex and rich wide area Internet telephony [18]. By embedding our connection establishment into the SIP phone call session setup, we need not design proprietary protocol for session context setup during the phone call setup between two clients. The whole connection establishment process is depicted in Figure 4.9.

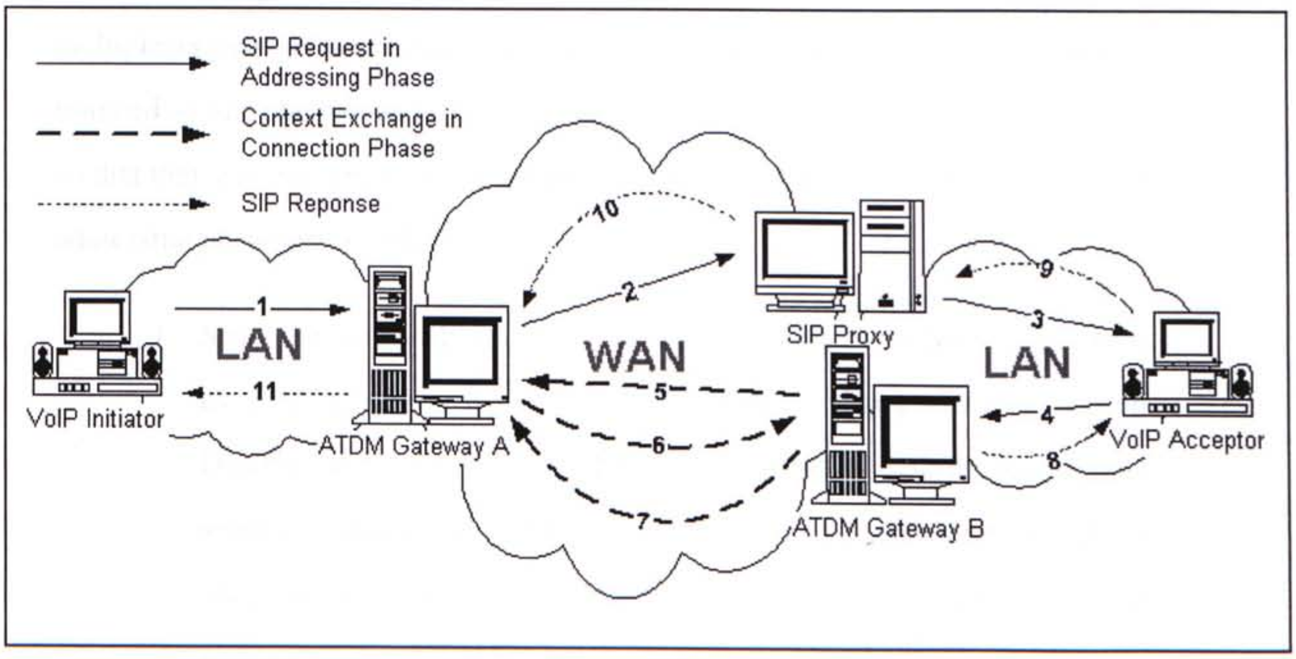


Figure 4.9: Connection Establishment Procedures.

4.4.1 Addressing Phase

In order to implement a scalable architecture, a multiplexer has no fixed list of "connectable" demultiplexers and we need not preset the list of demultiplexer beforehand. Instead, when the client initiates a connection with another client, the addresses of multiplexer and demultiplexer have to be located first. This is called the addressing phase.

Analogous to the relationship between a Web browser and its proxy server, every voice-over-IP application can set its own "gateway", which is defined as a server with both ATDM multiplexer and demultiplexer services running.

Both gateways do not know each other addresses when a sender on the multiplexer network wants to connect to the target receiver. An addressing phase is initiated so that one gateway can obtain the address of the gateway on the other side so that they can exchange context information in the connection phase. The steps in addressing phase are as follows:

1. Sender creates a SIP invitation request for a target user SIP address such as `sip:user@example.com`. This request also contains a Session Description Protocol (SDP) [19] as its content to describe multimedia session details of the sender like supported audio codecs, opened port for receiving audio RTP packets and IP address. This request is send to its preset gateway, ATDM Gateway A.

2. Gateway A looks up `example.com` in the Domain Name Service (DNS) and obtains the IP address of a server, which handles SIP requests for this domain, SIP Proxy in Figure 4.9. The request is proxied there.

3. The SIP Proxy server consults its location service, which resolves the IP address where the user, `user@example.com`, is currently logged in, i.e. the IP address of VoIP Acceptor in the figure. The request is then proxied to the VoIP Acceptor.

4. If the VoIP application on the acceptor machine is preset with a local gateway, ATDM Gateway B in the figure, it will proxy the invitation request together with its own multimedia session details in SDP format to Gateway B when the user accepts the invitation.

When Gateway B receives the SIP invitation request proxied from the VoIP Acceptor, Gateway B can get the IP address of Gateway A through the Via field defined in the SIP because every proxy machine that sends the request onwards must add its own addition Via field which contains its network address. Also, through the SDP content of the SIP request, Gateway B can know the IP addresses and the sending and receiving port numbers of both the VoIP Initiator and the VoIP Acceptor. After Gateway B locates the IP address of Gateway A and captures the audio session information of the two VoIP clients, it can initiate the connection phase as described in the next section.

4.4.2 Connection Phase

After addressing phase completes, Gateway B can start initiate a session connection between the specified Gateway A and Gateway A so that the two gateways can multiplex the audio RTP packets from their clients and can demultiplex the multiplexed packets from each other to their clients. Through the session connection, the context information is exchanged between the two gateways.

Conformed to the figure, the procedure for setting up a session connection between the two gateways is as follows:

5. Gateway B generates a session context identifier (CID_2) for audio traffic from VoIP Acceptor to VoIP Initiator and makes a connection request to Gateway A with CID_2 and the addresses and the sending and receiving port numbers of both the VoIP Initiator and the VoIP Acceptor.

6. After Gateway A receives the connection request, Gateway A generates another session context identifier (CID_1) for audio traffic from VoIP Initiator to VoIP Acceptor and creates a context entry of CID_2 for Gateway B. A demultiplexing port number will be assigned on Gateway A for the ATDM channel from Gateway B. Gateway A then sends back an CID_2 acknowledgement packet to Gateway B. The CID_1 identifier, the addresses and the sending and receiving port numbers of both the VoIP Acceptor and the VoIP Initiator are embedded into this acknowledgement packet.

7. When Gateway B receives the acknowledgement packet, it creates a context entry of CID_1 for Gateway A. A demultiplexing port number will be assigned on Gateway B for the ATDM channel from Gateway A. Gateway B then sends back an CID_1 acknowledgement packet to Gateway A.

After successful connection phase, Gateway B will send a SIP success response message back to VoIP Acceptor (8). This SIP success response will be sent through the proxy chain (9), (10), (11) back to VoIP Initiator. After VoIP Initiator gets the SIP success response, the two VoIP applications can send audio RTP packets to their corresponding gateways. The two gateways can multiplex the audio RTP packets with other connected RTP streams between two localities identified by the CIDs of different streams.

4.5 Software Implementation

As this scheme can be deployed by using application-layer protocol, we partially implemented the multiplexing and demultiplexing gateway for some basic testings on delay performance of multiplexing and demultiplexing services. The gateway software is based on UNIX environment. The gateway can multiplex small-sized timestamp packets into a multiplexed packet and send it to another gateway. The gateway program running on another machine can demultiplex the packets and send the packets to the corresponding receivers. However, the connection establishment processing and SIP integration have not been added into this software prototype.

By using this software, we can see the delay performance of multiplexing and demultiplexing services. It can be found that the delay performance of demultiplexing service is less than 1ms which is negligible in the end-to-end delay budget of interactive voice communications. For multiplexing service, the delay is mainly due to the waiting delay time, which corresponds to the time needed to wait for the arrival of a group of packets to be multiplexed. The total delay caused by other processing such as table lookup is less than 1 ms. So, it can be concluded that the processing time due to multiplexing and demultiplexing services is negligible.

This software prototype can show the possibility of implementing this scheme in application layer. A complete gateway can be implemented in the future based on this software prototype by adding the connection establishment communication protocol and realizing the full version of context mapping tables.

Chapter 5

Simulation Results

5.1 Simulation Model

The model used in the following simulations consists of *N* senders and *N* receivers. *N* senders in the same local area network produce voice traffic to *N* different receivers located in another local area network. All traffic flows through a common WAN link, which interconnects the senders' local area network and receivers' local area network. Our goal is to investigate how the proposed scheme performs under this scenario and compare the results with conventional schemes.

Each sender is connected to the multiplexer via an infinitely fast link and the multiplexer is connected to the demultiplexer via a relatively low bandwidth WAN link. Each receiver is also connected to the demultiplexer via an infinitely fast link as the same case in sender side. There is no propagation delay over any of these three links since our principal purpose is to investigate the delay imposed by sender, multiplexer and demultiplexer. Also, there is no blocking associated with the receivers. In other words, it is assumed that the receivers have infinite capacity for consuming voice packets.

For each packet sent by senders, RTP is used as the middle layer protocol for real-time information.

The network model used for the conventional schemes is shown in Figure 5.1 where μ is the WAN link service rate. The frame start times of senders are evenly distributed within the inter-departure time of audio RTP packets.

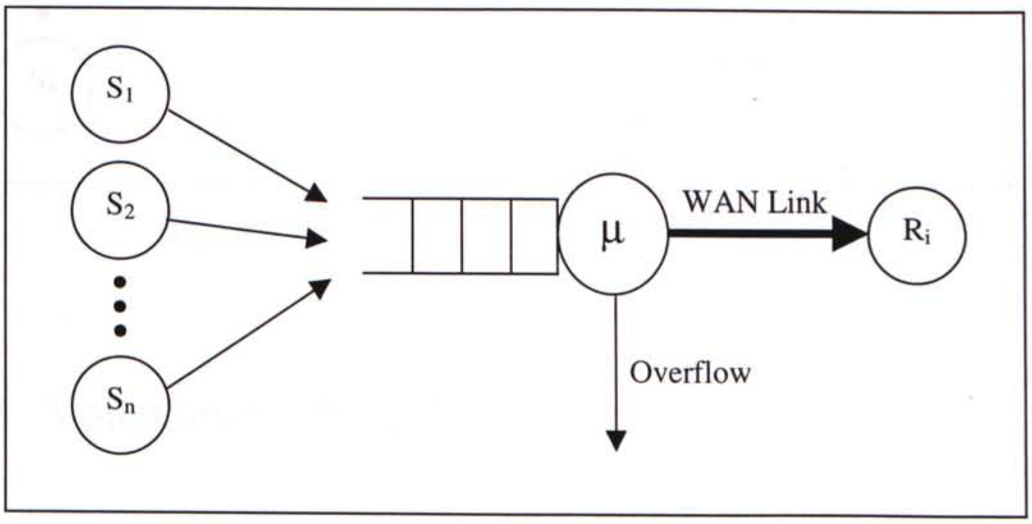


Figure 5.1: Network model of conventional schemes.

The network model used for our ATDM scheme is shown in Figure 5.2, where μ is the WAN link service rate. The frame start times of senders are also evenly distributed within the inter-departure time of audio RTP packets, which is one audio frame time when using our ATDM scheme. *M* is the multiplexer which multiplexes the audio 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 frame size of the WAN link. So, the multiplexer, *M*, will gives variable delay to every RTP packets from senders.

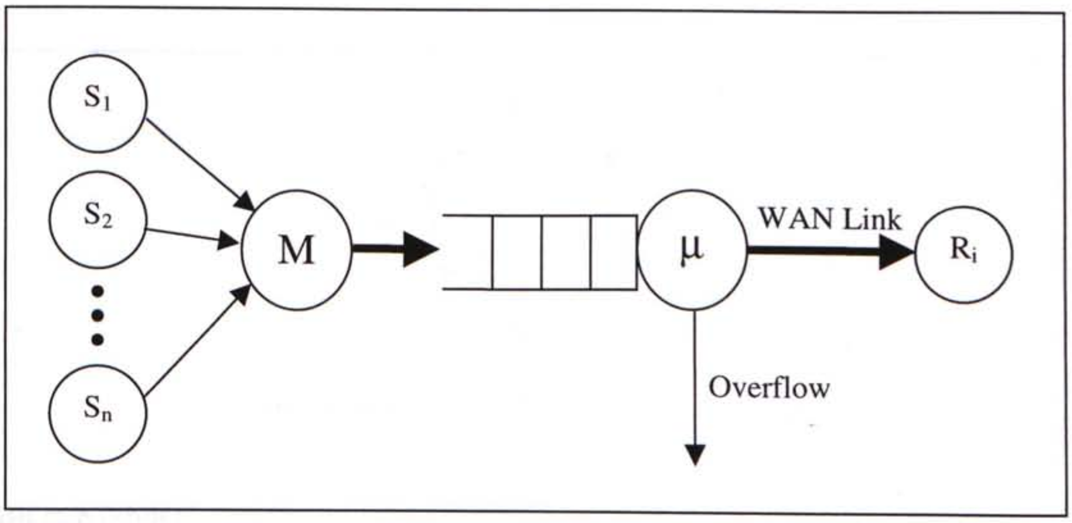


Figure 5.2: Network model of our ATDM scheme.

5.2 Voice Source Model

In the simulations, we assume that voice silence suppression is activated, and therefore, each voice source is modeled as a simple two-state variable-bit-rate ON-OFF source. This type of traffic source is used extensively to model standard voice sources [20]. The audio frame stream in each source consists of departures with T ms intervals. When a source is in the ON state, it will generate audio frames of size D_1 and when it is in the OFF state, audio frame size will be D_2. The ON and OFF state durations are distributed exponentially with means $1/\alpha$ and $1/\beta$ respectively. We adopt G.729A audio compression standard as the codec, so $T = 10$ ms, $D_1 = 10$

bytes and $D_2 = 4$ bytes. Also, we set $1/\alpha$ to 352 ms and $1/\beta$ to 650 ms. The voice source model can be visualized by Figure 5.3:

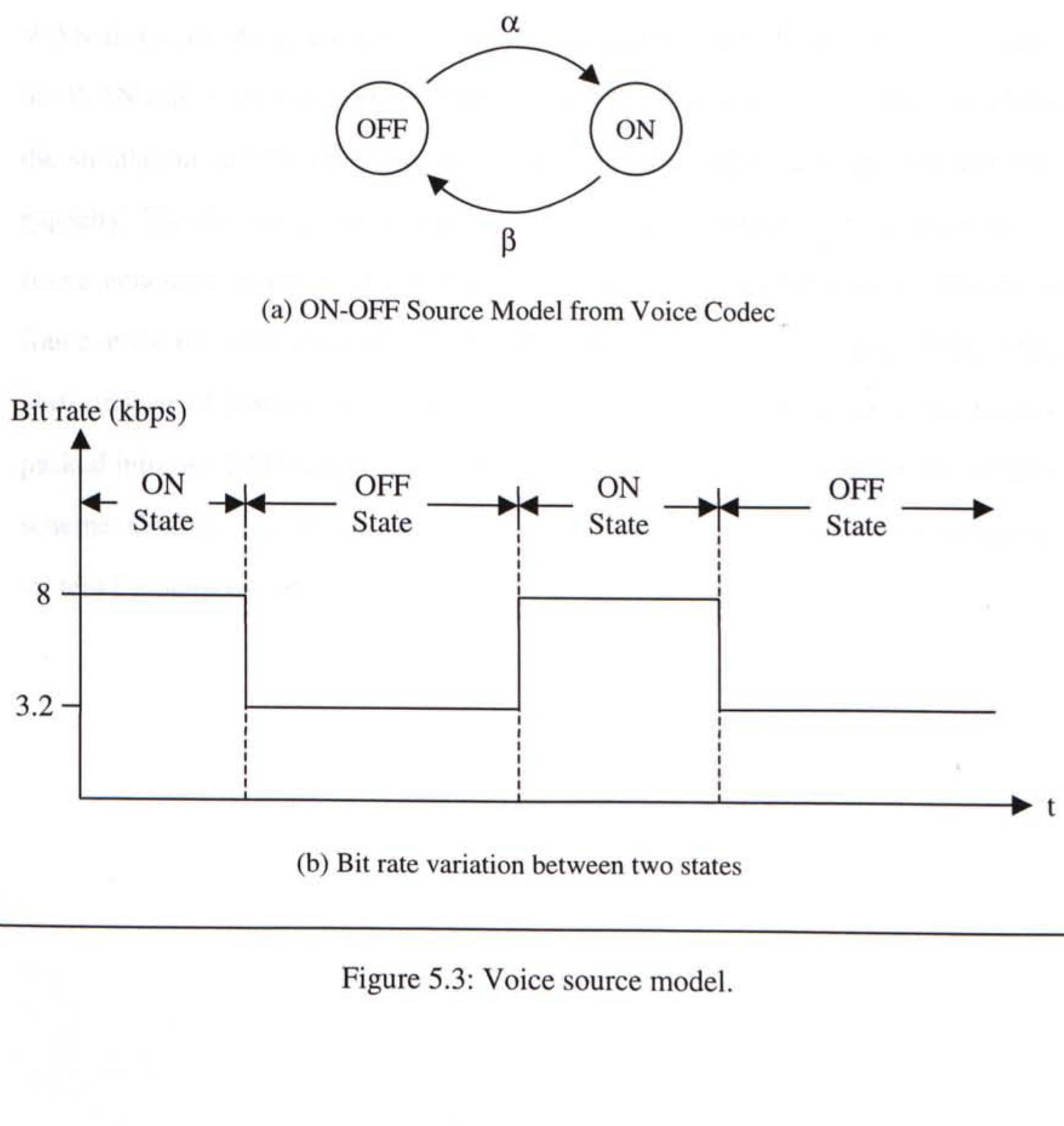

α
β
OFF
ON
(a) ON-OFF Source Model from Voice Codec
Bit rate (kbps)
ON State
OFF State
ON State
OFF State
8
3.2
t
(b) Bit rate variation between two states

Figure 5.3: Voice source model.

5.3 Simulation Results

5.3.1 Network Utilization and Delay Performance

Figure 5.4 depicts the average overall delay with zero transmission time over the WAN link utilization. We take the WAN link bandwidth to be 512 kbps. We vary the WAN link utilization by increasing the number of simultaneous connections *N* to the simulation and the utilization is calculated by the audio data rate over the link capacity. The overall delay shown in the figure is the beginning time of an audio frame generated at the sender to the arrival time of the RTP packet of this audio frame at the receiver. There are seven curves in the figure. Curves d1 to d6 show the performance of conventional schemes with the specified number of audio frames packed into one RTP packet. Curve dmux illustrates the performance of our ATDM scheme. Confidence intervals are shown for each point to indicate the time when the system becomes unstable.

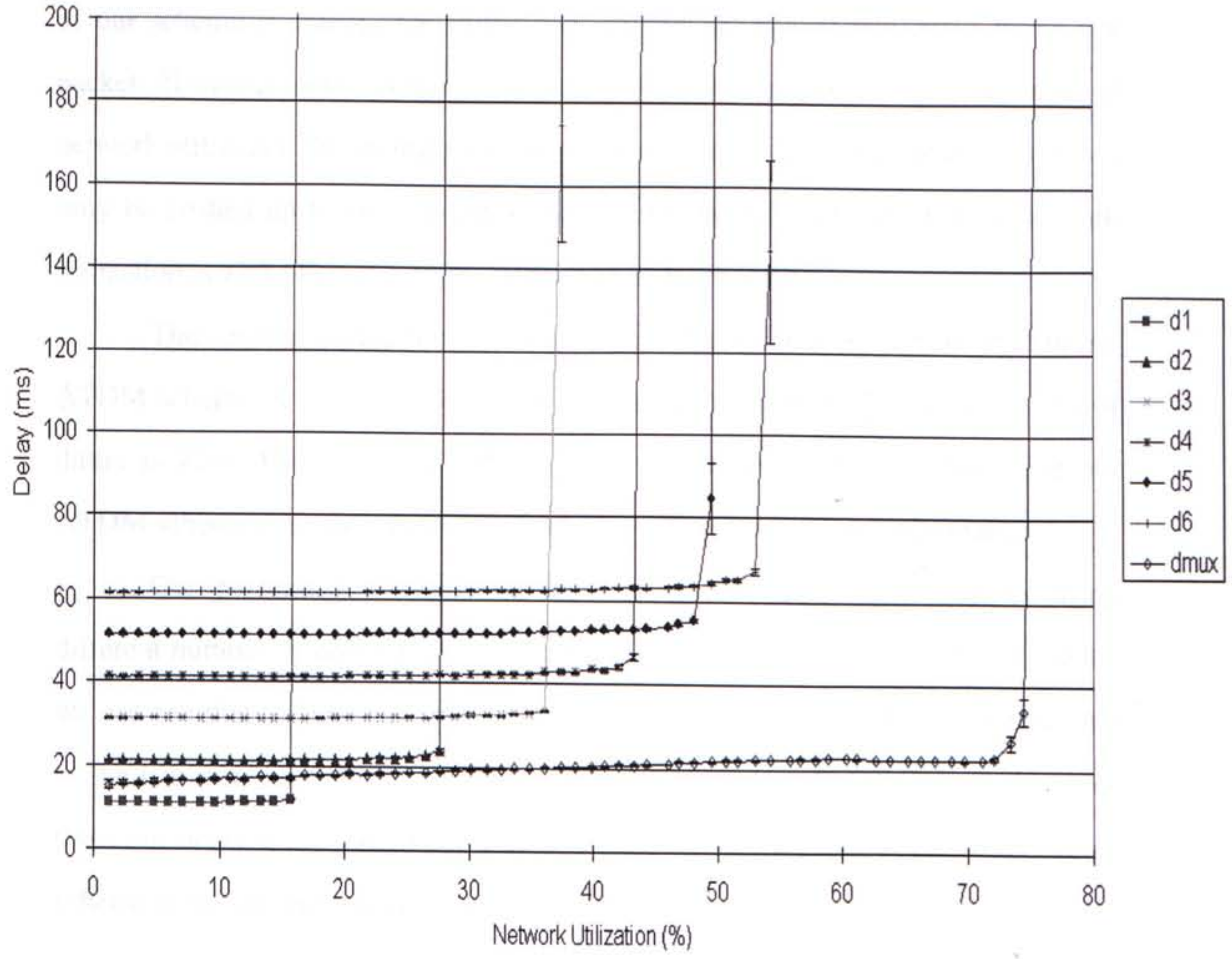


Figure 5.4: Comparison of the overall delay and the WAN link utilization between conventional schemes and ATDM scheme.

For conventional schemes, the average overall delay includes the assembly delay from sender and the queuing delay at the bottleneck of the WAN link. The average overall delay in our scheme consists of the assembly delay, the waiting time at the multiplexer, and the queuing delay at the WAN link bottleneck.

By comparing the curve d1 and dmux, we can see 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 16%. When using our ATDM scheme, the limit of network utilization is 72%. This is a 350% increase in network utilization.

The conventional scheme d2 is the one with similar delay performance to our ATDM scheme. It can be seen that the network utilization of d2 is 28% but that of dmux is 72%. This is still a 160% increase in network utilization for using our ATDM scheme over the conventional scheme with similar delay performance.

Our performance simulation includes other conventional schemes using different number of audio frames packed in one packet. From the simulation results, we can see that the network utilization of our scheme (72%) is still better than that of the conventional scheme using six audio frames per packet (52%). At the same time, our delay is around 20 ms, which is much lower than that in the conventional scheme using six audio frames per packet (60 ms).

5.3.2 Number of Supported Connections

Figure 5.5 shows the number of supported audio RTP connections of different schemes over different WAN link bandwidth. We carried out a set of experiments on different WAN link bandwidths from 64 kbps to 1.536 Mbps. In each experiment, we varied the WAN link utilization by increasing the number of simultaneous connections *N*. When the delay variance jumps significant suddenly from around 10 ms^2 to more than 200 ms^2, this means that the system becomes unstable at this condition. Therefore, the number of supported audio RTP connections for each WAN link bandwidth N_{max} can be collected just before the delay variance at (N_{max} + 1) connections becomes significant (larger than 200 ms^2). There are also seven curves in the figure for comparison. Curves d1 to d6 present the number of supported audio RTP connections of conventional schemes with the specified number of audio frames packed in one RTP packet. Curve dmux illustrates the number of supported audio RTP connections of our ATDM scheme.

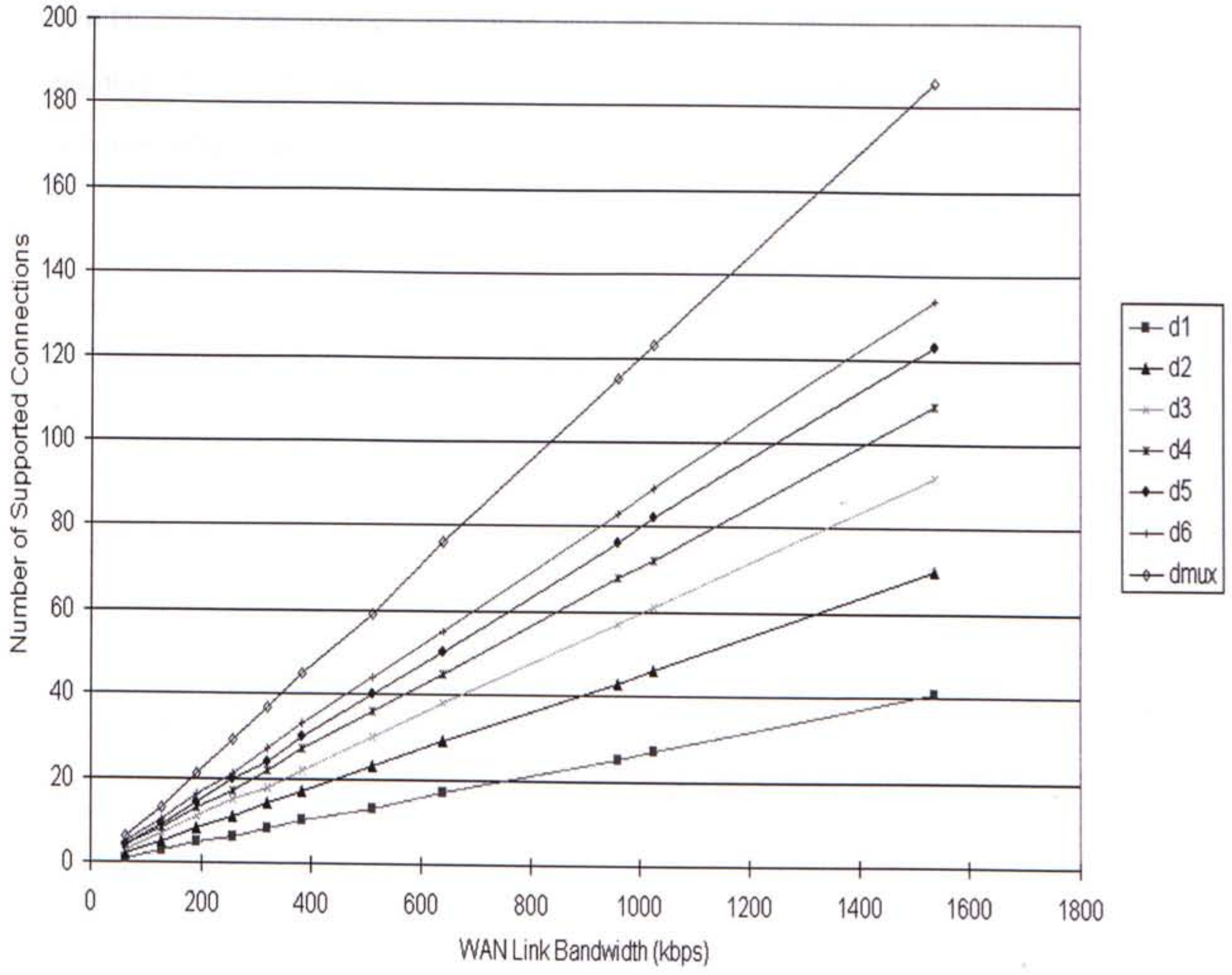


Figure 5.5: Number of supported connections of different schemes over WAN links of different bandwidth.

It can be seen that our proposed scheme can be applied to WAN links of different bandwidth without affecting its efficiency. This shows that the ATDM scheme can always outperform the conventional schemes in terms of number of supported connections. Specifically, it can always support 160% more audio connections than the two-audio-frame-per-packet conventional scheme, which is of similar delay performance compared with our ATDM scheme. The difference in number of supported connections between a conventional scheme and our proposed scheme will be much more significant as the WAN link bandwidth increases.

Chapter 6

Conclusion and Future Work

In this thesis, we have proposed a new Asynchronous Time Division Multiplexing (ATDM) scheme for voice-over-IP applications to improve the bandwidth efficiency while maintaining low overall end-to-end delay. In conventional voice-over-IP, a source waits for enough data to be sampled before sending them out as a packet at regular intervals. The data collection time can be quite long, contributing to the overall end-to-end delay. An alternative is to send out small packets containing a small amount of voice data each. However, this may result in bandwidth inefficiency due to the packet header, which is of fixed length regardless of the amount of data in it. This tradeoff is fundamental. Our ATDM scheme attempts to remove this inefficiency by means of aggregating voice traffic from several sources into shared IP packets.

With the prevalence of voice-over-IP, one would expect many simultaneous voice sessions between two localities. In this case, the simultaneous voice streams can be multiplexed to improve the bandwidth efficiency on the long-distance path between the two localities. Specifically, voice sources at a locality sends their packets to a multiplexer, where the headers of the packets are stripped off and replaced with smaller 2-byte mini-headers. A number of packets together with their mini-header are then put into a regular IP packet, co-shouldering the overhead of the shared header. The mini-header identifies the RTP stream and stores important information for recovering timing RTP fields and is used at a demultiplexer near the destination to reconstruct the original packet of the source before forwarding it to the destination.

We have investigated the performance of this ATDM scheme relative to that of the conventional voice-over-IP systems with the use of simulations. The results show that the proposed scheme can provide better bandwidth efficiency with insignificant increase in the overall end-to-end delay such that the WAN link can support more voice-over-IP connections at the same time.

We show in this thesis that our multiplexing scheme can reach 72% bandwidth utilization while conventional scheme with similar delay performance can only achieve 28% network utilization, corresponding to a 160% improvement. This means the number of connections over a WAN link can be greatly increased with the use of our ATDM scheme.

This ATDM scheme is flexible and can be applied to voice-over-IP systems using 2-state variable-bit-rate speech codecs. The scheme is also robust in that small amount of packet loss in the WAN link does not affect the performance.

For future work, the ATDM scheme can be extended to a multi-level aggregation system. That is, with enough traffic volume, one can construct a hierarchical network with several levels of multiplexing, much like the current plain old telephone service (POTS) network. In such a hierarchical network, audio RTP packets from a locality with the same next hop, instead of the same destination locality, can be multiplexed together. The next hop gateway will demultiplex the RTP packets and multiplex those who destine to the same next hop. This can increase the multiplexing possibility in a multiplexer so that it can achieve a higher multiplexing gain. However, this will introduce additional delay in every multiplexing hop and connection setup will be more complex. The benefits and shortcomings of this extension remain to be further studied.

Bibliography

[1] M. Hassan, A. Nayandoro, and M. Atiquzzaman, "Internet Telephony: Services, Technical Challenges, and Products", *IEEE Communications Magazine*, Apr. 2000, pp. 96 – 103.

[2] E. C. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture", *IETF Internet Draft*, draft-ietf-mpls-arch-06.txt, Aug. 1999.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", *IETF RFC 2475*, Dec. 1998.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification", *IETF RFC 2205*, Sept. 1997.

[5] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", *IETF RFC 1889*, Jan. 1996.

[6] ITU-T Recommendation G.723.1, "Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kbit/s", Mar. 1996.

[7] ITU-T Recommendation G.114, "One Way Transmission Time", Feb. 1996.

[8] P. T. Brady, "Effects of Transmission Delay on Conversational Behavior on Echo-Free Telephone Circuits", *Bell Labs Technical Journal*, vol. 50, 1971, pp. 115 – 134.

[9] B. T. Doshi, E. J. Hernandez-Valencia, K. Sriram, Y. T. Wang, and O. C. Yue, "Protocols, Performance, and Controls for Voice over Wide Area Packet Networks", *Bell Labs Technical Journal*, Vol. 3, No. 4, Oct. – Dec. 1998, pp. 297 – 337.

[10] ETSI GSM 6.10 version 5.1.1, "Digital Cellular Telecommunications System (Phase 2+) Full Rate Speech Transcoding", May 1998.

[11] ITU-T Recommendation G.729 Annex A, "Reduced Complexity 8 kbit/s CS-ACELP Speech Coder", Nov. 1996.

[12] R. V. Cox, "Three New Speech Coders from the ITU Cover a Range of Applications", *IEEE Communications Magazine*, Sept. 1997, pp. 40 – 47.

[13] S. Casner, and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", *IETF RFC 2508*, Feb. 1999.

[14] X. Xiao, and L. M. Ni, "Internet QoS: A Big Picture", *IEEE Network*, Mar./Apr. 1999, pp. 8 – 18.

[15] S. Wright, and R. Onvural, "IP Telephony vs. ATM: What is There to Discuss?", *Proceedings of the 1st IEEE International Conference on ATM*, June 1998, pp. 400 – 409.

[16] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol", *IETF RFC 2543*, Mar. 1999.

[17] H. Schulzrinne, and J. Rosenberg, "The Session Initiation Protocol: Providing Advanced Telephony Services across the Internet", *Bell Labs Technical Journals*, Vol. 3, No. 4, Oct. – Dec. 1998, pp. 144 – 160.

[18] H. Schulzrinne, and J. Rosenberg, "Signaling for Internet Telephony", *Proceedings of the Sixth International Conference on Network Protocol*, Oct. 1998, pp. 298 – 307.

[19] M. Handley, and V. Jacobson, "SDP: Session Description Protocol", *IETF RFC 2327*, Apr. 1998.

[20] K. Sriram, and W. Whitt, "Characterizing Superposition Arrival Processes in Packet Multiplexers for Voice and Data", *IEEE Journal on Selected Areas in Communications*, Vol. SAC-4, No. 6, Sept. 1986, pp. 833 – 846.