Order Number 9521325

Multimedia network synchronization in real-time applications

Zarros, Panagiotis Nikolaos, Ph.D.

City University of New York, 1995

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Multimedia Network Synchronization In Real-Time Applications

by

Panagiotis Nikolaos Zarros

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirement for the degree of Doctor of Philosophy, The City University of New York.

1995

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

11/21/94
Date:

Chairman of the Examining Committee:
Dr. Tarek N. Saadawi, Professor of Electrical Engineering, The City College of the City University of New York.

11/21/94
Date:

Executive Officer:
Professor Gerard Lowen, Dean of the Graduate Studies at the Engineering School, The City College of The City University of New York.

Dr. Myung J. Lee	Co-Mentor, Assistant Professor, Department of Electrical Engineering, The City College of the City University of New York.
Dr. Patrick L. Combettes	Associate Professor, Department of Electrical Engineering, The City College of NY.
Dr. Ibrahim W. Habib	Assistant Professor, Department of Electrical Engineering, The City College of NY.
Dr. M. Umit Uyar	Associate Professor, Department of Electrical Engineering, The City College of NY, Distinguished Member at AT&T Bell Labs.
Dr. Nenad M. Marinovic	Associate Professor, Department of Electrical Engineering, The City College of NY.
Dr. Dhadesugoor Vaman	Professor, Director of Advanced Telecom Institute, Dept of Electr. Eng. and Comp. Science, Stevens Institute of Technology.
Dr. Kazem Sohraby	Distinguished Member at AT&T Bell Laboratories

The City University of New York

Abstract

Multimedia Network Synchronization In Real-Time Applications

by

Panagiotis Nikolaos Zarros

Advisor: Professor Tarek N. Saadawi

Various synchronization issues arising in the transfer of voice, video, or data through communication networks for real-time applications are the subject of this thesis. Multimedia communication involves many complex problems that do not appear in other means of communication, such as radio communication. This is because in multimedia communication networks, each type of traffic may travel independently from the other and experiences different network jitter (variation of the delay). This becomes further complicated by the absence of a universal clock.

The multimedia synchronization transport protocol presented in this thesis is divided into three building blocks: a) Point-to-point single-medium synchronization block,, b) Intermedia synchronization block, and c) Interparticipant synchronization block.

The proposed multimedia synchronization protocol is based on statistically evaluating the reference times or expected arrival times of the packets from each participant in each connection. Once the reference times are estimated, synchronization is achieved not according to the generation times of the packets but according to their expected arrival times. This approach exhibits the optimum delay, because packets from each

connection exhibit the maximum delay seen only in their own connection. Furthermore, deletion/addition of each participant and/or media connection can happen independently to other participants and media connections.

An effort was also devoted in representing the sequence number of each packet with the minimum number of bits. It is proven that the time interval the optimal sequence numbering scheme for multimedia synchronization should span needs not to cover a time duration greater than the maximum of the following two expressions: the maximum variation of the network jitter and twice the maximum absolute difference between the expected time delays of the two media.

Another issue is the frequency offset existing between the sender and the receiver. Knowledge of the frequency offset implies accurate knowledge of the generation period T^s of the packets at the sender, and this in return implies a more accurate multimedia synchronization. The frequency offset estimation is based on evaluating the tangent among two subsets of the arrived packets which are constructed so they lead to minimum mean square error.

Dedication

My Thesis is dedicated to the memory of my father
Nikolaos Ioannis Zarros,
and to my mother
Aikaterini Panagiotis Photopoulos — Zarros.

Also, my thesis is dedicated to all poor people who dared and left their homes to study abroad, i.e. they took the long and hard journey to Ithaka, irrespectively whether they "succeeded" or not because "And if you find her poor, Ithaca has not defrauded you. With the great wisdom you have gained, with so much experience, you must surely have understood by then what Ithacas mean", and therefore are the only ones who can really judge and understand what I went through to reach ITHACA.

And finally, I would like also to dedicate my thesis to my Philology teacher in the Lykeion of Argos Orestikon,
Mr. George Kales,
for his teachings had a tremendous influence on my character.

Acknowledgments

At some time, Alexander the Great was speaking very proudly for his teacher Aristotelis. Then, someone from the audience questioned him as follows: "Tell me Alexander, If you were not Alexander, whom you would prefer to be like? Your father, Philip, The King of Macedonians and of all the Greeks, or your teacher Aristotle, the greatest philosopher of our times? " And Alexander gave the following famous reply: "To my father I owe that I am alive, but to my teacher I owe that I live well."

The answer given by Alexander the Great synopsizes the importance that teachers may play in the life of a person. Anything anybody achieves in life is a matter of his own convictions and sacrifices, but it is also directly related to the help, influence, and encouragement, he receives from the outside world. Similarly, I will also like to acknowledge the help and support I received from a number of people that made possible for to me to finish this Thesis.

First of all, I would like to thank my mentor Professor Tarek N. Saadawi, for his great help, encouragement, and support, during the years I was his student. I have been impressed by his character, his generosity, his honesty, and his judgement. Without him, I would have not been able to finish my Ph. D.

Also, thanks to Professor Lee, who has been my co-mentor, for his help in my work.

I would like also to thank Professor Nenad M. Marinovich who had been my previous mentor. I am referring especially to the fact that he is the one who hired me as a Unix System Administrator for the Electrical Engineering Department. Without this job my dream to finish my Ph. D would not have been realized. But also for the many things that I learned from him. Also, I would like to thank Professor Donald L. Schilling, for his inspired teachings in Electrical Engineering has given me a good understanding of many issues of Electrical Engineering.

In addition, I would like to thank all the members of my Ph.D. committee, Professors T. Saadawi, M. Lee, P. Combettes, I. Habib, M. Uyar, N. Marinovich, D. Vaman, and Kazem Sohraby for taking the time to review my thesis and making valuable comments to this work. Also, my thanks to Dean Gerard Lowen, who has been the Dean of the Graduate studies for most of the time I was a graduate student, for helping me in several occasions.

I would like also to thank my friends Beth R. and Sonia V., for they gave me a lot of moral support during difficult times. Without their support, I would not had the courage to finish my studies.

Also, I would like to thank Mr. George Kales, my Philology teacher in High School (Lykeion), for his teachings had a tremendous impact on my character. I thank him a lot.

Finally, there are two Institutions that I owe many things to them. The Lykeion of Argos Orestikon, Greece, which in my time, 1978-1981, was ranked among the best High Schools in the Northern Part of Greece. This was due to two teachers: Mr. Kebaptsoglou (died in 1984), and Mr. G. Kales, both of them philology teachers. The other Institution is of course the City College of New York. At the City College of New York I spent eleven (11) years studying. Without the City College of New York being here, I would not have had the opportunity to go to College.

In the next pages, few poems from the many that Mr. George Kales had analyzed in the class and I found them extremely valuable, follow.

Ithaca

When you start on your journey to Ithaca,
then pray that the road is long,
full of adventure, full of knowledge.
Do not fear the Lestrygonians
and the Cyclopes and the angry Poseidon.
You will never meet such as those in your path,
if your thoughts remain lofty, if a fine
emotion touches your body and your spirit.
You will never meet the Lestrygonians,
the Cyclopes and the fierce Poseidon,
if you do not carry them within your soul,
if your soul does not raise them up before you.

Then pray that the road is long.
That the summer mornings are many,
that you will enter ports seen for the first time
with such pleasure with such joy!
Stop at Phoenician markets,
and purchase fine merchandise
mother-of-pearl and corals, amber and embony,
and pleasurable perfumes of all kinds,
buy as many pleasurable perfumes as you can;
visit hosts of Egyptian cities,
to learn and learn from those who have knowledge.

Always keep Ithaca fixed in your mind.
To arrive there is your ultimate goal.
But do not hurry the voyage at all.
It is better to let it last for long years;
and even to anchor at the isle when you are old,
rich with all that you have gained on the way,
not expecting that Ithaca will offer you riches.

Ithaca has given you the beautiful voyage.
Without her you would never have taken the road,
But she has nothing more to give you.

And if you find her poor,
Ithaca has not defrauded you.
With the great wisdom you have gained,
with so much experience,
you must surely have understood by then
what Ithacas mean.

Constantine P. Cavafy,
Alexandria, Egypt, 1863–1933

Thermopylae

Honor to those who in their lives
are committed and guard their Thermopylae.
Never stirring from duty;
just and upright in all their deeds,
but with pity and compassion too;
generous whenever they are rich, and when
they are poor, again a little generous,
again helping as much as they are able;
always speaking the truth,
but without rancor for those who lie.

And they merit greater honor
when they foresee (and many do foresee)
that Ephialtes will finally appear,
and in the end the Medes will go through.

Constantine P. Cavafy,
Alexandria, Egypt, 1863–1933

Che Fece . . . IL Gran Rifiuto

To certain people there comes a day

when they must say the great Yes or the great No.

He who has the Yes ready within him

reveals himself at once, and saying it he crosses over

to the path of honor and his own conviction.

He who refuses does not repent. Should he be asked again,

he would say No again. And yet that No-

the right No- crushes him for the rest of his life.

Constantine P. Cavafy,
Alexandria, Egypt, 1863–1933

A Solitary Swallow; The Axion Esti

A solitary swallow and a costly spring,
For the sun to turn it takes a job of work,
It takes a thousand dead sweating at the Wheels,
It takes the living also giving up their blood.

God my Master Builder, You built me into the mountains,
God my Master Builder, You enclosed me in the sea!

Magicians carried off the body of May,
They buried the body in a tomb of the sea,
They sealed it up in a deep well,
Its scent fills the darkness and all the Abyss.

God my Master Builder, You too among the Easter lilacs,
God my Master Builder, You felt the scent of Resurrection!

Wriggling like sperm in a dark womb,
The terrible insect of memory breaks through the earth
And bites the light like a hungry spider,
Making the shores glow and the sea radiant.

God my Master Builder, You girded me with seashores,
God my Master Builder, You founded me on mountains.

Odysseus Elytis, The Axion Esti,
Athens, Greece, 1911–

Contents

List of Figures

Chapter 1 Introduction

1.1 Multimedia Synchronization: Definition, Motivation, Issues.

Technological advances in computers and communications make it feasible today to speak about real-time interaction with other people using images as well as voice. One such class of applications is multimedia teleconferencing, which permits individuals to interact real-time in a conference-like manner by exchanging audio, video and text information. Another class is the so-called video-store type of applications. It is envisioned in the future that people will be able to use their computer monitor for entertainment purposes, such as watching a movie. In that case the user will not go to the video store to pick up the proper tape, but through the computer network he/she will download and playback in real-time the movie in his computer monitor. This type of application can also be utilized where the user wants to scan images and text from a library (*i.e.* a book), or see what is in sale on the local supermarket. We make a distinction about multimedia conference type of applications and video-store type of applications such that in the former case both sides interact with each other in real-time, while for the video-store type of applications, only one side is acting in real-time. A reader interested in the fundamentals of networking should refere to the book by Saadawi *et al.* [1].

One inherent problem in multimedia, as the word multimedia etymologically implies, is associated with inter-mixing different media into one. Here the problem of *intermedia synchronization* arises. This intermedia synchronization deals with synchronizing the temporal relations among different media; an example of this is lip-synchronization. There is another type of synchronization to be considered. If a conference is not in its simple form with only two participants but in its more general form with three or more participants, management issues and synchronization problems with packets (messages)

arriving from different participants arise. Here the synchronization problem will be referred to as *interparticipant synchronization*.

The structure of the introduction is as follows: In the next section, our experimental work which is related to the construction of an actual multimedia conference system and making experiments is briefly presented (this was a team effort; many students of the Computer Communications Laboratory were involved in this project). In Section 1.3, a synopsis of the proposed multimedia synchronization transport protocol is presented. In Section 1.4, we introduce the reader to the significance of the estimation of the frequency offset in real-time network applications, and specifically to multimedia synchronization. Finally, in Section 1.5, the contents of this thesis are summarized.

1.2 Experimental Work Performed.

It is generally accepted that most of the currently existing transport protocols such as *TCP/UDP* and *OSI* transport protocols are not suitable for multimedia conference applications, in which different media services require different service requirements. However, the wide use of these protocols make them attractive to people who are interested in finding the performance of these protocols and quickly bring multimedia to the desktop. For these reasons, we have designed, realized and analyzed a multimedia conference system based on the test-bed *FDDI* network existing in the Computer Communications Laboratory using *TCP/IP* and *UDP/IP* protocols. End-to-end delays and jitters are measured to compare the performances of two architectures: one architecture is based on *TCP* in which all application services, interactive video, interactive voice and data, are multiplexed at the application level and transmitted through *TCP*; the other architecture has interactive video and voice transmitted through *UDP* and data transmitted through *TCP*.

During the various experimentation we conducted, we found a way to measure the time delay which avoids the synchronization problem caused by the frequency offsets existing between the clocks at the sender and at the receiver. Essentially, this was an idea that was applied to the estimation of the time delays of the packets by exploiting the bi-directional nature of the traffic generated in a multimedia conference. Before that, estimating the average time delays of the packets passing through a given computer network route was an extremely complicated matter. One other reason which justifies our involvement in designing an entire new multimedia conference system was to get a better understanding of the issues involved, and to share our experience with others by publishing the architecture of our multimedia conference system. Although many software companies have been involved in designing a multimedia conference system, no publication has appeared in the literature. However, our design of our multimedia conference system has already been published [2].

The large bandwidth available from the fiber results in a shift of the system bottleneck from channel bandwidth to communication protocol processing. Many researchers have investigated on the performance of current transport protocols [3, 4, 5, 6, 7]. For example in [3], a comparative evaluation of *TCP*, *UDP* and *VMTP* protocols is carried out and a conclusion is made that the existing transport protocols are not suitable for future multimedia high speed networks, since they fail to achieve the level of performance in terms of on-time reliability and jitter control. Research on software architecture, protocol and conference management for multimedia is reported in [7, 8, 9, 10, 11]. In many studies, however, the software architecture issue is only dealt with on an abstract level, without going into the actual implementation. In contrast, our work presents an actual implementation addressing problems arising both from the software architecture and the synchronization among processes in a software for multimedia services. For a detailed

analysis of our multimedia conference software, the reader is referred to Chapter 3.

1.3 The Multimedia Synchronization Transport Protocol.

1.3.1 Motivation: The Interparticipant Synchronization Problem.

What prompted us to develop a new multimedia synchronization transport protocol was the synchronization problem we encountered in the most complex case, the case of *Interparticipant Synchronization*. Interparticipant synchronization is related to synchronization of packets arriving from different sources. As shown in Figure 1.3.1, receiver R can be any of the participants or a separate mixer and as noted both the average time delay D^i and the network jitter are different from participant to participant. Although we will define exactly the term jitter later, it suffices for now to define jitter as the variability of the time delay that packets experience when passing through a network. This model is in agreement with the real situation. Due to many factors such as specific geography of each connection, different routes taken, or different network load (how high the utilization of a certain network link is), make out the network model appearing in Figure 1.3.1 for a multimedia conference the correct model. This model is an improvement from previous models used by other researchers [12, 13, 14], where the time delays for all connections in a multimedia conference are considered to be constrained within the *same bounds* of the time delay.

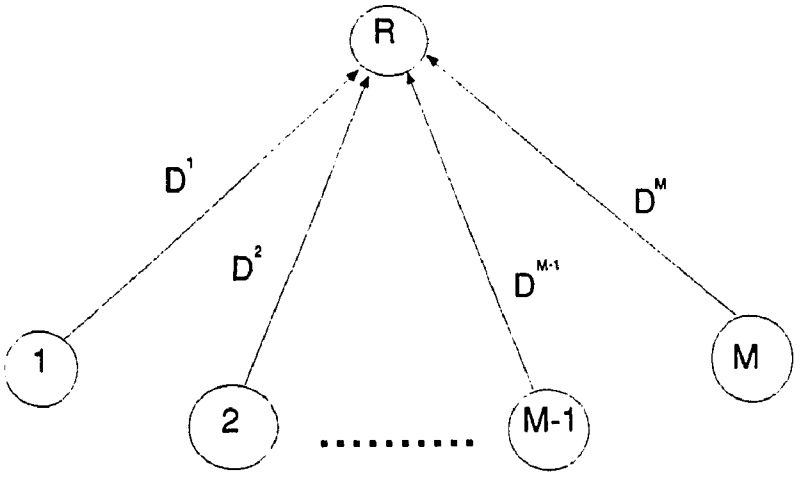

Figure 1.3.1 Multi party conference with M+1participants.

In our approach, the existence of different bounds on the arrival time of a packet from different sources is recognized to improve the synchronization algorithm. Whenever a reference to packets is made we really mean the messages from the application level. In the paper authored by Rangan *et al.* [12] (their synchronization algorithm has been proven to be optimum in the case there is no exchange of any control messages), the deterministic approach is followed whereas the statistical approach is taken in this thesis. A similar problem, the problem of voice synchronization between sender and receiver, has been studied by many researchers [15, 16, 17, 18].

Interparticipant synchronization is key to a multimedia multi-party conference. Applications of a multimedia conference such as coediting of a document or mixing of different voice streams require the maximum possible synchronization of the arriving packets. There are two sources of problems affecting interparticipant synchronization: firstly, the absence of a global clock because of the fractional drifts among different clocks (frequency offsets), and secondly, the existence of variability in the arrival time of packets from the same source. The aforementioned problems should be solved in a

manner that the buffer requirement at the receiver and the total delay for playback of the received packets are minimum.

Our research in this field has resulted in an idea about synchronization in multi-party multimedia conference that leads to an optimum solution *w.r.t.*. the time delay. To further illustrate what is meant by an optimum solution, imagine a multimedia conference held by three participants. One participant is located at the City College, the second in New Jersey and the third in California. Let us suppose that we are concerned only with the synchronization at our site, *i.e.* we want to synchronize packets arriving at City College from New Jersey and California. Synchronization between packets generated from different sources is better illustrated with voice packets. If we want to simultaneously listen to many participants, then before playing back the voice packets at the receiver, the voice packets from the different participants have to be added and only then are to be played back. In all previous works done by other researches, the interparticipant synchronization problem has been solved by delaying every packet according to the maximum delay observed from all the connections in the multi-party conference. In other words, packets generated from New Jersey will have to be delayed the same amount of time as packets generated from California. However, we have observed that, for purposes of multimedia synchronization, packets generated from New Jersey need to be delayed only the maximum delay observed in the connection from New Jersey to the City College.

While realizing this idea about interparticipant synchronization, a number of new ideas and concepts were introduced that we believe will be used as the backbone theoretical tools for a research in the field of multimedia synchronization, either it is interparticipant, or intermedia, or simply receiver-sender synchronization. An important point here is the concept of the reference times or simply the expected arrival times. In other words, a reference time is the hypothetical arrival time that a packet would have,

if the packet coming from the sender to the receiver would have experienced exactly the average time delay for this connection. Another fundamental idea is the observation that if at some point in time synchronization was achieved, and by some way we were also able to exactly determine the frequency offsets between the sources and the receiver, then synchronization is to be achieved for the rest of the time without requiring to reexecute the synchronization algorithm. Yet, the most important issue is that the estimation of the reference times and of the frequency offsets does not need even a single exchange of a feedback packet between the communicating entities.

The synchronization algorithm is also shown to encompass cases where traffic sources transmit packets with different periods. This ability of the proposed multimedia synchronization algorithm, *i.e.* to synchronize packets that are generated from sources with different periods, will be shown to be very useful for the specific case of intermedia synchronization. The reason is that, for the general problem of intermedia synchronization, the voice sources might transmit packets with one period and the video sources with another period. Then the problem of intermedia synchronization becomes simply a problem of interparticipant synchronization where sources transmit packets with different periods.

The degree of accuracy of synchronization depends on how accurate the estimation of the reference times are. Modeling the probability density function of packet arrivals with a Rayleigh, Normal and Uniform density functions, an upper bound in the probability of error derived using the Chernoff bound is plotted with respect to the number of received packets N. For instance, by allowing an error of 0.15 of Δ_{max} (where Δ_{max} is the maximum jitter packets are experiencing passing through one connection) only 25 packets are required for $P(\epsilon) = 0.05$ using the Rayleigh model. Using simulations, the same performance can be achieved with only 15 packets. This means that at the connection/setup time, workable synchronization can be achieved in a very short time,

in the neighborhood of one second. This displays the high feasibility of the proposed algorithm for practical applications, because we would not like to wait for more than a second during the connection/setup time. In this work no effort has been devoted in modeling the probability density function (*p.d.f.*) of the packet arrival times. A reader interested in the theoretical analysis leading to the proper *p.d.f.* should refere to [19].

1.3.2 The Three Building Blocks of the Multimedia Synchronization Protocol.

The solution to interparticipant synchronization led our research to propose a three-layer concept to be used for multimedia synchronization. In other words, the fact that packets from each connection exhibit only the maximum delay seen in their own connection, (independent of the delay existed to the other connections), led us to the introduction of the multimedia synchronization transport protocol as presented in this thesis. The multimedia synchronization transport protocol /algorithm, as shown in Figure 1.3.2, is divided into three blocks:

1. Point-to-Point Single-Medium Synchronization Block,
2. Intermedia Synchronization Block, and
3. Interparticipant Synchronization Block.

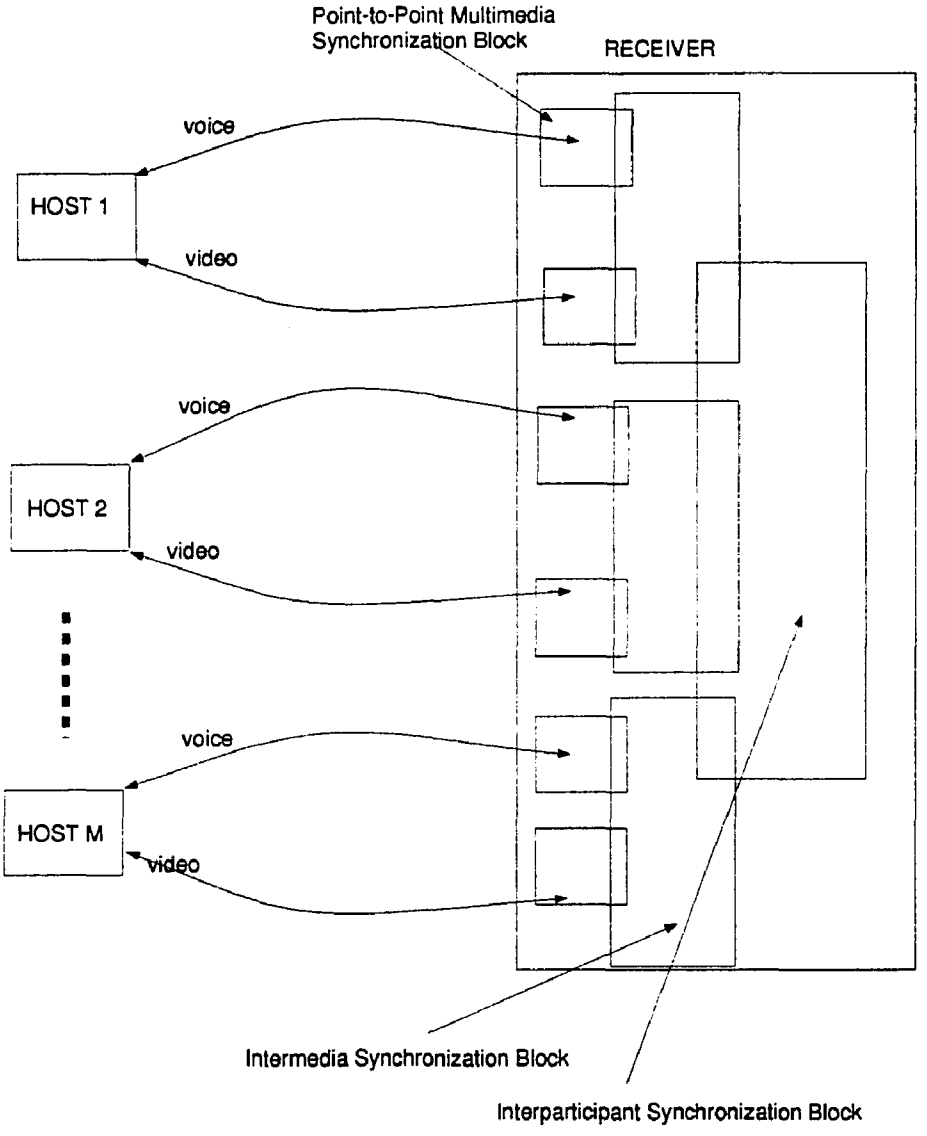

Figure 1.3.2 The three building blocks of the Multimedia Synchronization Transport Protocol.

The relationships among the three building blocks are shown in Figure 1.3.2. The proposed multimedia synchronization protocol is based on statistically evaluating the reference times or expected arrival times of the packets from each participant in each connection. Once the reference times are estimated, synchronization is achieved not according to the generation times of the packets but according to their expected arrival times. This concept has many benefits. In particular permits the receiver to decouple

the synchronization dealing with packets arriving from one participant from the synchronization dealing with packets arriving from the other participants. In addition, for the same reason it exhibits the optimum delay. To furhter clarify these points, suppose a multimedia conference is held among three participants. One of them is located in New York, another in New Jersey, and the third in California. Then packets arriving from New Jersey to the participant located in New York, will only experience the maximum time delay expected in the connection from New Jersey to New York. Similarly, packets arriving from California will only experience the maximum time delay expected in the connection from California to New York. This has as an end result that the case of interparticipant synchronization can logically be broken into three parts.

It follows that the first building block deals with the synchronization of the packets arriving from a single participant for a single connection (only one medium). The second synchronization block deals with intermedia synchronization. In this thesis when a reference is made to intermedia synchronization we will mean synchronization between video and voice packets only, a case most widely known as lip-sync. In other words, the functionality of the Intermedia synchronization block is to make certain that the temporal relations between the packets of the two media are kept during playback at the receiver. An additional assumption about the intermedia synchronization block is that the generation periods T of the packets for the two media can be different but the period of the voice packets must be an integer multiple to the generation period of the video packets, i.e. $T^{vi} = zT^{vo}$. Finally, the third synchronization block deals with interparticipant synchronization, where packets from different participants are required to be mixed (for the case of voice packets) or displayed (for the case of video packets) at the receiver.

The difference between interparticipant and intermedia is that in interparticipant, the *phase offsets* are unknown but the transmission periods of the sources are the same,

while in intermedia the phase offsets are zero but packets are generated from sources with different periods. Phase offsets exist when media sources residing in different hosts try to communicate with another host. Then, even if the periods are the same for all the hosts, the starting point will be different for each of the sources. For intermedia synchronization, since media sources (video and voice) reside at the same host, one can arrange so that a video frame is being tautochronously transmitted every m'^{th} voice packet transmitted, where m = 1, 2, 3, etc. Then, no phase offset problem exists.

We would like to note here that the proposed protocol can also work with the future high speed networks, such as *ATM* networks. In such networks, the variability of the time delay (network jitter) will be very small, and the sequence numbering of the packets will be taken care by the underlying *ATM* layers. In that case, evaluation of the reference times will happen much faster, and multimedia synchronization will follow according to the theory developed.

1.3.3 Optimal Sequence Numbering Scheme

In this thesis, we also paid attention in finding a solution to multimedia synchronization problems that uses the minimum number of control bits. Actually, we have proven that the time interval Ψ the optimal sequence numbering scheme for multimedia synchronization (this is valid for all three blocks: Point-to-Point, Intermedia, and Inter-participant) should span, needs not to cover a time duration greater than the maximum of the following two expressions: the maximum variation of the network jitter and twice the maximum absolute difference between the expected time delays of the two media. The number of the bits $p_\psi = \lceil \log_2 \frac{\Psi}{T} \rceil$ is shown to be a function of the sampling frequency used and the coding. Therefore, in the case of sampling frequency of 8kHz and 8-bit *PCM* coding, *i.e.* the quality of voice in a multimedia conference application is the same

quality that we have in our current telephone conversations, then it is proved that the optimum number of bits p_ψ needed for multimedia synchronization can be as little as three (3). This is a significant result, and it implies that a number of people, say five (5), can talk and stop talking independent of one another, but when they talk again, their packets are to be immediately synchronized by simply having the sender transmit three bits along with each voice packet.

1.4 Frequency Offset Estimation in Real-Time Network Applications.

Transfer of real-time of data, among other things, imposes strict requirements on time delay and synchronization. The last requirement, due to the packet switching technology employed in computer networks, has led many researchers in developing algorithms/protocols for network timing synchronization [20, 21, 22, 23, 24]. Nonetheless, the nature of Network Timing Synchronization leads to complicated algorithms that make the implementation of such an algorithm/protocol questionable by many researchers in the field. An alternate approach to network timing synchronization is proposed in this thesis. Clock (timing) synchronization is not anymore the responsibility of the network itself but that of the participants communicating in real-time using the computer network.

Network timing synchronization provided to a group of hosts only when they are communicating with each other alleviates many of the burdens existed for a "global" or "regional" network timing synchronization. Much of the complexity of such an approach arises from the requirement that good operation is needed also in the presence of a node failure. This is especially true for a point-to-point communication where a node failure will directly be translated as the end of the connection. Another requirement adding to the complexity of a "global" network timing synchronization protocol is that such a protocol tries to provide the network with a global or universal clock, in the sense that the

clocks of each host try to show the same time at any time. However, with the concept of the reference times or expected arrival times that we have introduced, a universal time is no longer necessary for real-time applications. This is due to the concept of the reference times because now packets from different participants become synchronized not according to their generation times but according to their expected arrival times. Thus, only estimation of the frequency offset is needed and not of the actual time offset existing among the different hosts. Since different hosts are geographically distributed, propagation delays are different in each connection complicating any synchronization protocol trying to estimate the time offset.

For this reason, in this thesis a simple, cost-effective, near-optimal with respect to the mean square error criterion algorithm for estimating the frequency offset in real-time applications is presented. We would like to emphasize that the operation of the proposed algorithm is based solely on the packets exchanged between the two users and no extra packet transmission is required. The frequency offset is assumed to be a linear function of time. In reality, as denoted by Mitra in [20] and Mills in [21], for time intervals in the order of micro-seconds the frequency offset is a non-linear function of time. However, for real-time network applications the inter-arrival times of the packets is in the order of tens of milli-seconds resulting in cancellation of the non-linearities.

The frequency offset estimation is based on evaluating the tangent among two subsets of the arrived packets which are constructed so they lead to minimum mean square error. The estimation is further improved by combining the different estimations done locally at each host. The coupling of the estimations from different hosts provides adequate estimation to the frequency offset also to the participant who has not received any packets, or very few, during the time interval in consideration. The computational cost of the proposed algorithm is significantly lower than the least squares approach,

while its mean square error is only a fraction (in the simulation example only 1%) higher than the optimum.

1.5 Structure of the Thesis.

In Chapter 2 (see also [25]), an overview of related work done by other researchers is summarized, and in Chapter 3, we present our work with respect to building the multimedia conference system. In Chapters 4, 5, and 6, the three building blocks of the multimedia synchronization transport protocol are presented. Specifically, in chapter 4 we discuss the point-to-point single-medium synchronization block, in chapter 5 the intermedia, and in chapter 6 the interparticipant synchronization block. In chapter 7, we present our algorithm for estimating the frequency offset between the clocks at the sender and the clock at the receiver. Finally, in Chapter 8 we give our concluding remarks.

Chapter 2 Overview of Work Related to Multimedia Synchronization Performed By Other Researchers

2.1 Introduction

One of the problems arising in transporting multimedia information over computer networks is multimedia synchronization. Multimedia synchronization plays an even more significant role in the computer networks in contrast to the data communications, *i.e.* radio, telephone and mobile communications, because of the packet switching environment of the computer networks. Communications in computer networks is achieved with the transportation of packets, that may be of fixed size like in *ATM* [26], or of a variable size like ethernet. Each packet carries in a reserved place (control field) informational bits that will permit it to transverse the network and find its destination independently. The time delay each packet experiences to travel from the source to the destination is variable depending on many uncontrollable factors, such as: the specific route the packet followed, the number of gateways (routers or hosts which sit on the boundary of two networks) the packet traveled through, the load (number of currently active processes) of the gateway at the time the packet was passing through, the state and the capacity of the network buffers (queues) of each gateway, the processing speed of the gateway, the capacity of each link, *etc.*.

A fundamental problem associated with multimedia synchronization in a packet switching environment is the *network jitter*. Network jitter is the variability of the time delay each packet experiences traveling though a network. Although the phenomenon of network jitter is known to the researchers in the field since late 70's, different definitions of the network jitter exist and used by different researchers [27, 28, 29]. An additional

problem that one encounters in multimedia synchronization is the absence of a universal clock. In theory, as shown in Figure 2.1.1, a global clock can be achieved using appropriate protocols/algorithms that distribute the time in all the hosts of the network. For more details about the operation of those protocols, the interested reader is referred to Mills [21] and Mitra [20]. However, providing the entire network with a universal clock is a fairly complex task, especially in the case of a failure in one or more of the master hosts. Master hosts are the hosts recognized by all the hosts in the network as the ones having their clocks closest to the correct time. An alternative solution to the absence of universal clock for real-time applications is proposed in [30, 31]. Clock (timing) synchronization is not anymore the responsibility of the network itself but that of the participants communicating in real-time using the computer network.

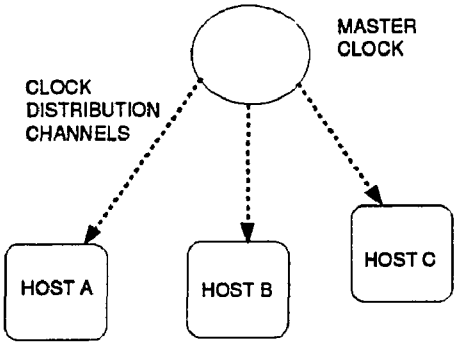

Figure 2.1.1 Global network clock.

In this chapter, different multimedia synchronization issues and their solutions are presented. The purpose of this chapter is not to give a detailed description of the various algorithms, but rather to provide the reader with a comprehensive understanding of the different synchronization problems existed in multimedia communications, and present in a nutshell some of the most important works in the field.

The organization of the rest of the chapter is as follows: in section 2, different definitions of the network jitter given by different researchers is provided and an analysis is made with respect to their utilization. In section 3, techniques for packet voice synchronization are described, while in section 4 algorithms for interparticipant synchronization are presented. Section 5 deals with synchronization protocols for the efficient transport of multimedia data over computer networks, and finally in section 6 we present our concluding remarks.

2.2 Analysis of the Network Delay, Jitter and Burstiness

As will be shown later in section 3, for any type of multimedia synchronization, one is interested not in the average time delay but in the maximum time delay. Packets are played back (voice packets) or displayed (video packets) according to the maximum delay seen at the connection. Thus, it is imperative for multimedia synchronization, or more generally, for any real time network application which demands minimum delay in the playback of the packets, that the maximum delay is minimized.

A pioneer work in the theoretical analysis of the network delay has been made by Cruz in [27]. In this paper theoretical tools for obtaining bounds on the delay are presented. He assumes that the data stream entering the network satisfies "burstiness constraints". He shows that even if a traffic does not satisfy such constraints, the traffic can be shaped to satisfy any prescribed burstiness constraints by passing it through a network element called regulator.

Suppose that the rate of the traffic stream entering a network element is *R(t)*, and the network element process data at the rate ρ. Then, for any $t_2 \geq t_1$, $\int_{t_1}^{t_2} R(t)$ is the amount of data from the stream that is transmitted on the link in the time interval $[t_1, t_2]$. The burstiness constraints are defined as follows. Given $\sigma \geq 0$ and $\varrho \geq 0$, we write

$R \backsim (\sigma, \varrho)$ if and only if for all t_1, t_2 satisfying $t_2 \geq t_1$ there holds

$$\int_{t_1}^{t_2} R dt \leq \sigma + \varrho(t_2 - t_1) \quad . \tag{2.2.1}$$

A useful interpretation of the constraint $R \backsim (\sigma, \varrho)$ can be gained by defining the backlog function $W_\varrho(R)$ as follows:

$$W_\varrho(R) = \max_{t_2 \geq t_1} \left[\int_{t_1}^{t_2} R dt - \varrho(t_2 - t_1) \right] . \tag{2.2.2}$$

$W_\varrho(R)$ is the size of backlog, *i.e.* the amount of unfinished work, at time t_2 in a work conserving system, which accepts data at a rate described by the rate function *R(t)* and transmits data at the rate ρ. This constraint can be generalized if we consider that for a function b, $R \backsim b$ if and only if

$$\int_{t_1}^{t_2} R(t) dt \leq b(t_2 - t_1) \ , \quad t_2 \geq t_1 \ . \tag{2.2.3}$$

Note that if b is defined as $b(t) = \sigma + \varrho t$, then $R \backsim b$ whenever $R \backsim (\sigma, \varrho)$.

Define the delay experienced by a data bit passing through a network element as the difference between the time when the given data bit enters the element and the time when the bit exits the element. Let us also represent the maximum delay for the given traffic stream passing through the element as $\bar{D}$. Then, if the input traffic R_{in} conforms to some function b_{in}, *i.e.* $R_{in} \backsim b_{in}$, then the output traffic R_{out} conforms to $R_{out} \backsim b_{out}$, where $b_{out}(t_1) = b_{in}\left(t_1 + \bar{D}\right)$. The proof is as follows

$$\int_{t_1}^{t_2} R_{out} dt \ \leq \int_{t_1 - \bar{D}}^{t_2} R_{in} dt \ \leq b_{in}\left(t_2 - t_1 + \bar{D}\right) = b_{out}(t_2 - t_1) \ . \tag{2.2.4}$$

The first inequality comes from the fact that in a work conserving system, the amount of traffic entering the system equals the amount of traffic exiting. By bounding the time

interval of the entering traffic with the worst case situation, *i.e.* $t_1 - \bar{D}$, we get the inequality. This is an important result because it lets us characterize the burstiness of the traffic exiting a network element which introduces a maximum delay $\bar{D}$, given that we know the characteristic of the input traffic.

A similar approach is followed by Wang and Crowcroft in [29]. There, the traffic is characterized by a pair of parameters (ρ_{max}, ρ_{min}). The traffic is divided into smaller units, called synchronization units. Suppose that there are M packets in a synchronization unit and let a_i bet the arrival time of the i^{th} packet. If $A(t_1, t_2)$ denotes the number of packets arrived in the interval $[t_1, t_2)$, then $A(a_1, a_m)$ is the number of packets arrived before $t = a_m$, and $A(a_1, a_m) = m - 1$. Let $X_m = \frac{A(a_1, a_m)}{a_m - a_1}$ and $Y_m = \frac{1}{a_m - a_{m-1}}$, $m = 2, 3, ..., M$. Then, ρ_{max}, ρ_{min} are defined as follows:

$$\begin{aligned} \rho_{min} &= \min [X_2, X_3, ..., X_M] \\ \rho_{max} &= \max [Y_2, Y_3, ..., Y_M] \ . \end{aligned} \tag{2.2.5}$$

The traffic of the synchronization unit conforms to (ρ_{max}, ρ_{min}), if $X_m \geq \rho_{min}$, $Y_m \geq \rho_{max}$ for $m = 2, 3, ..., M$. We can interpret ρ_{max} to be the peak rate of the incoming traffic and ρ_{min} to be the highest rate at which a server is never idle.

The burstiness of the m^{th} packet is defined as

$$b_m = a_1 + \frac{A(a_1, a_m)}{\rho_{min}} - a_m \ . \tag{2.2.6}$$

If the incoming traffic is served at a rate ρ_{min}, then $a_1 + \frac{A(a_1 + a_m)}{\rho_{min}}$ is the time when the server starts to process the m^{th} packet. Thus, b_m is the amount of time that the m^{th} packet has to wait in the queue before it can be served, while the traffic is served at a fixed rate ρ_{min}. We can view $a_1 + \frac{A(a_1 + a_m)}{\rho_{min}}$ as the *expected arrival time* and b_m as the difference between the actual arrival time and the expected arrival time.

In [29], the delay experienced by the first packet is the target delay. If d_m is the time when the m^{th} packet departs from the network, then the jitter is defined as follows:

$$j_m = (d_m - a_m) - (d_1 - a_1) \ . \tag{2.2.7}$$

Suppose that the server is a work conserving and the flow is served at rate $\rho \geq \rho_{min}$. A packet has arrived or departed only when its last bit has arrived or departed. In other words,

$$d_m = \max\left[d_{m-1}, a_m\right] + \frac{1}{\varrho} \ , \ \ m = 2, 3, ..., M \ . \tag{2.2.8}$$

The delay experienced by the first packet depends on the backlog of packets left over the previous synchronization unit. Suppose that D is the time that the first packet has to wait before being served. Then for the first packet we have

$$d_1 = D + \frac{1}{\rho} + a_1 \ . \tag{2.2.9}$$

Using the above definitions, the authors are able to prove the following. If the input traffic conforms to (ρ_{max}, ρ_{min}) and $\rho \geq \rho_{min}$, then the output traffic conforms to (ρ, ρ_{min}). Thus, this is another way to characterize the output traffic of a network element when the input traffic conforms to some burstiness constraints. Similarly, the authors are able to bound the network jitter j_m of the m^{th} packet as follows:

$$j_m \leq (m-1)\left(\frac{1}{\rho} - \frac{1}{\rho_{max}}\right) \ . \tag{2.2.10}$$

This bound allows us to calculate the minimum bandwidth for a given jitter constraint. Suppose J_{max} is the maximum jitter the receiver can handle. The minimum bandwidth

required can be derived from

$$J_{max} \leq (M-1)\left(\frac{1}{\rho} - \frac{1}{\rho_{max}}\right) \Rightarrow \rho = \frac{(M-1)\rho_{max}}{J_{max}\rho_{max} + (M-1)} \quad . \tag{2.2.11}$$

In the papers just described, the main objective was to analyze the phenomenon of the network jitter in relation to burstiness in order to minimize the maximum delay encountered in the network. In other words, they look at the problem form the network designer's point of view. In [32, 33, 34], a less theoretical approach is followed whose objective is that given a network, what are the most effective theoretical tools that can be utilized in order the temporal relations among different data streams existed during their generation times to also be preserved during playback at the receiver. More about this definition can be found in later chapters.

The definition of the network jitter given in this Thesis is essentially identical to the definition given in Matragi *et al.* in [28]. The interarrival times between successive packets are assumed to be independent. Let I_n denote the interarrival time between the n^{th} and the $(n+1)^{th}$ packet, and let D_n denote the interdeparture time. Then the jitter is defined as the sequence

$$\widetilde{J} = \{J_n : \ n \geq 1\} = \{D_n - I_n : \ n \geq 1\} \ . \tag{2.2.12}$$

There is however a difference in the functionality between them. Equation (2.2.12) requires knowledge of the inderdeparture times of the packets, in other words the generation times of the packets, while using our definition, the arrival times of the packets are sufficient for the determination of the network jitter. This is a fundamental result that it will permit the designer of multimedia synchronization algorithms to lower

the complexity of the synchronization algorithms, to eliminate feedback packets used for the estimation of the network jitter, and decuple the synchronization of packets from one participant from the packets arrived from the other participants.

2.3 Techniques for Packet Voice Synchronization

The fundamental problem that we are addressing in this section is to reconstruct a continuous stream of speech packets that arrive with varying time delay. In other words, as shown in Figure 2.3.2, a sender transmits periodically packets to the destination. If we were to construct the probability density function of the packet arrivals at the receiver, the *p.d.f.* would look very similar to the one shown in Figure 2.3.2.

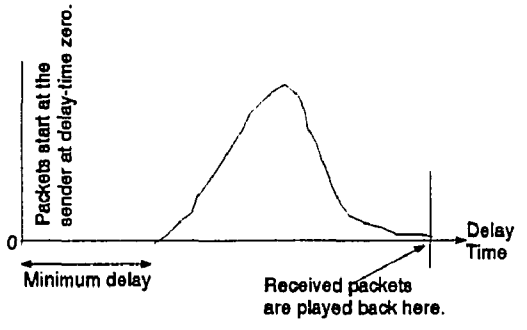

Figure 2.3.2 Packets are played back according to the maximum delay observed.

The playback time of the packets will be this time where most of the packets are assumed to have arrived. For instance, if N is the total number of packets arrived at the receiver, the playback time $T_{playback} = T_D^{max}$ might be defined as the time that 99.9 % of the received packets (or $0.999 \times N$) were arrived within the time interval $\left[\min_n \{T_D\}, T_D^{max}\right]$, $n = 1, 2, ..., N$.

Barberis and Pazzaglia in [15] were among the first that addressed the problem of packet voice synchronization. Much of their work is devoted into an analysis of the statistics of the packets arriving at the receiver, and at the end they propose an algorithm

for sender-receiver synchronization. The algorithm works as follows. For each first packet of every talkspurt the birth time stamp is coded in the header according to the time base of the transmitter clock. This value is read by the *PVR* (Packet Voice Receiver) with another time base, and thus an error delay estimate is introduced; however, if the gap is removed, the *PVR* knows the exact transit time.

The time base of the transmitter clock starts when the first packet of the call is played out. The time stamp coded in the header of the first packet of the first talkspurt is set at zero. When this packet joins the *PVR*, its clock is switched on, taking into account the deterministic delay (transmission and switching times) experienced in the network path by the packet; this means that the stochastic delay is assumed to be zero. In this way the gap between the two clocks equals the random delay of the first packet of the first call. The time stamps of the following packets of the first talkspurt are read by the *PVR* according to the time base of its clock. In this way whenever the random delay of a packet lower than the random delay of a previous one, the time clock of the receiver assumes a value lower than the possible one, and so, a clock correction operation is made.

W. Montgomery summarizes the various techniques for packet voice synchronization in [16]. The same approach followed in [15] is followed here as well. The synchronization of the sender (called *PVS*, packet voice sender) and the receiver (called *PVR*, packet voice receiver) are synchronized after an estimation of the time delay. He presents four ways of estimating the time delay:

Blind Delay. Here the worst case assumption of the time delay is used. The first packet arrived in *PVR* is assumed to have experienced the minimum delay, and therefore its playback time will be after the maximum variable delay (jitter) a packet is expected to experience passing through the network.

Round trip Measurement. The measurement is made by sending a packet containing

a local clock reading from the *PVS*. When the packet arrives at the *PVR*, it is immediately sent to the *PVS*. When it arrives at the *PVS*, the *PVS* calculates the roundtrip delay by subtracting the clock value in the packet from the current time. The roundtrip delay is then sent to the *PVR*, and all subsequent packets sent by the *PVS* to the *PVR* contain timing information relative to the first packet sent by the *PVS*.

Absolute Timing. Here the clocks at the *PVR* and *PVS* are synchronized to the same absolute time reference. Each packet carries an indication of its generation time, and the *PVR* uses that to compute the target playback time. The timestamp need not be capable of encoding the absolute time, but must encode time to a sufficient resolution to allow the *PVR* to unambiguously determine when the packet was produced. If the *LSB* represents 1ms, the 8 bits are enough for a maximum variation of the delay (jitter) of 250ms.

Added Variable Delay. Here, the actually variable delay that each packet experiences as it occurs is measured. The variable delay measurement can be made by carrying a "delay stamp" indicating the accumulated delay in each packet. Each network element adds its delay to the delay stamp as it passes through. The process used by the *PVR* to determine the target playout time is as follows: a) Subtract the delay stamp from the local clock to determine the time at which the packet was produced, plus any fixed delays not encoded in the delay stamp. b) The playout target time is then determined by adding the maximum expected delay stamp value to produce the generation time of the packet plus the maximum expected transit delay.

2.4 Interparticipant Synchronization (Media-Mixing) Algorithms

In section 3, synchronization algorithms for packets (voice) belonging to the same stream/channel were presented. In this section we present algorithms that deal with the synchronization of packets emanating from two or more participants. A direct application

of those algorithms arises in teleconferencing applications such as *teleorchestra*. Because in many of the applications packets from different participants are required to be mixed at the receiver, the first researchers who dealt with the problem (Rangan, Vin, and Ramanathan in [12, 13, 14]) referred to these algorithms as *media mixing* algorithms.

In [12, 13], synchronization of packets is achieved at the receiver according to the generation times of the packets. However, there is no need for transferring a timestamp or any other extra bits in order for the generation time to be inferred. Basically, their approach is as follows. The time delay of any packet in any connection in a given multiparticipant conference is bounded by the minimum and maximum time delays that packets can possibly have in a given conference. This implies that when a packet arrives at the receiver, an inference can be made for its generation time. For example, if a packet from source i arrives at time t_n^i, then we can infer that its generation time $g(t_n^i)$ is within the time window $g(t_n^i) - D_{max} \le g(t_n^i) \le g(t_n^i) - D_{min}$. The generation time for the next $n+1$ packet from the same source i will be

$$g(t_{n+1}^i) - D_{max} \le g(t_{n+1}^i) \le g(t_{n+1}^i) - D_{min} \quad and \quad g(t_{n+1}^i) = g(t_n^i) + T^i \ . \quad (2.4.13)$$

Similarly for the other sources. Using these relations, the authors have constructed logical rules that are used to infer the proper generation time of a packet. However, there are some disadvantages of their algorithm. The algorithm is not working under all circumstances, *i.e.* the algorithm cannot determine the proper set of packets to be mixed at all times. This is due to the fact that when the network jitter $D_{max} - D_{min}$ is larger than the generation period T of the packets, there is no way one can always determine the proper generation times of the packets without using some sort of sequence numbering. Secondly, if there is one participant with large maximum time delay, then packets generated from the other participants will also experience the maximum time delay of this participant during playback.

A synchronization algorithm which solves the problems of the algorithm described above is shown in [32, 34, 35], and in chapter 6. There, packets from different participants are synchronized according to their expected arrival times (the reference times) and not according to their generation times.

2.5 Muliparty-Multimedia Synchronization Protocols

In section 3, the issue of transporting real-time data between sender and receiver was addressed, while in section 4 solutions to the problem of synchronizing packets from different participants were described. In this section, protocols for synchronizing packets/objects of multimedia type and from many participants are described. However, the interest of these protocols is not in describing the means by which the frequency offset, time offset, or network jitter, can be estimated, but they mostly concentrate on defining the synchronization actions to be taken. In other words, given certain *QoS* requirements of the application, they give the playout time for each packet. Whether or not packets belong to the same multimedia segment (packets belonging to the same multimedia segment have to be played back "together"), this is assumed to be achieved from the algorithms described in previous sections (3 and 4).

Significant work in multimedia synchronization has been accomplished by Little and Ghafoor in [36, 37], and later continued by Woo and Ghafoor in [38, 39]. Specifically in [36, 37] the authors present a two level synchronization architecture. The lower level, called the network synchronization protocol (*NSP*), provides functionality to establish and maintain individual connections with some specified synchronization characteristics. The network delays can be characterized by either measurement techniques, as presented in [16], or by guaranteed quality of service offered at access to the network (connection establishment). The *NSP* provides a general interface and functionality for many

applications. The upper level, the application synchronization protocol (*ASP*), supports an integrated synchronization service for multimedia applications. Contrary to *NSP*, *ASP* provides specific services pertaining to the retrieval of complex multimedia objects from multiple sources across a network for playout at a single site for the application at hand.

The temporal specifications of a multimedia object is done using a form of Petri net called object composition Petri net (*OCPN*) [37]. The *OCPN* is defined as a bipartite directed graph, specified by the tuple (see Figure 2.5.3) $\{T, P, A, D, R, M\}$ where: T is a set of transitions, P is a set of places, A is a set of directed arcs, D is a mapping form the set of places to the real numbers (time durations), R is mapping from the set of places to a set of resources, and M is a mapping from the set of places to the integers.

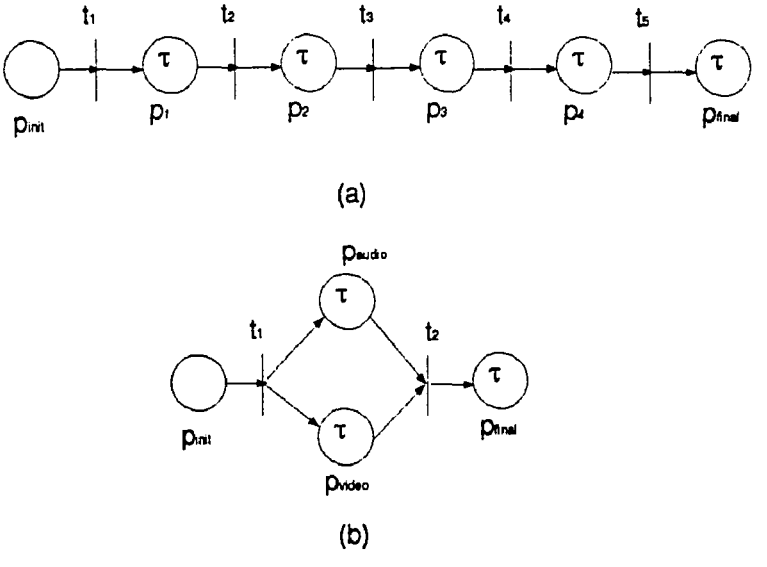

Figure 2.5.3 (a) Sequential OCPN, (b) Concurrent OCPN [36, 37].

The overall processes needed for the implementation of *ASP* are as follows. 1) Retrieval of the temporal relations describing the components of the complex multimedia object, 2) evaluation of the precedence relations of the *OCPN*, thereby creating a playout schedule, and 3), decomposition of the schedule into subschedules based upon the

different traffic classes represented. The *NSP* protocol takes as input a set of objects, their playout times, their start time, and the desired probability of synchronization failure. Then, the *NSP* protocol provides us with the following. 1) A predicted end-to-end control time T for the set of the data elements, 2) a point-to-point data transport mechanism, and 3) signaling from destination to source to control playout rate.

Li *et al.* in [40] present a multimedia segment delivery scheme (*SDS*) for the simultaneous delivery of multimedia data. *SDS* employs the synchronization quality of service (*SQoS*) parameters to guarantee the simultaneous delivery of the different types of data. The *SQoS* are the maximum tolerable skew parameters that a user accepts to exist between the different types of data. For instance, if a mismatching greater than 100ms between voice and video data is considered unacceptable, then 100ms will be the skew tolerance parameter between the voice and the video data. The proposed *SDS* algorithm performs synchronization control and maintains the required coupling by limiting the synchronization errors within the corresponding skew tolerance parameters.

The multimedia segment is defined as the synchronization granularity of the multimedia data. A segment contains multiple data streams. Streams within a segment are supposed to be displayed simultaneously at the receiver. The desired segment length is constrained by the required *SQoS*. $\Delta\tau_{i,j}$ is defined as the skew tolerance parameter between the data streams i and j. Define t_i and t_j to be the average delay between the display of the two adjacent packets belonging to stream i and j respectively. Let $T_i(T_j)$ be the time slot of one i (j) packet. Then the segment size S (time length) is constrained by

$$\begin{cases} \max\{T_i, T_j\} \leq S \leq \dfrac{T_i T_j \Delta\tau_{i,j}}{T_i t_j - T_j t_i} & if \quad \left(\dfrac{T_i}{T_j} > \dfrac{t_i}{t_j}\right) \\ \max\{T_i, T_j\} \leq S \leq \dfrac{T_i T_j \Delta\tau_{i,j}}{T_j t_i - T_i t_i} & if \quad \left(\dfrac{T_i}{T_j} < \dfrac{t_i}{t_j}\right) \end{cases} \tag{2.5.14}$$

which is derived by the following constraints

$$S\left|\frac{t_i}{T_i} - \frac{t_j}{T_j}\right| \leq \Delta\tau_{i,j} \quad and \quad S \geq \max\{T_i, T_j\} \; . \tag{2.5.15}$$

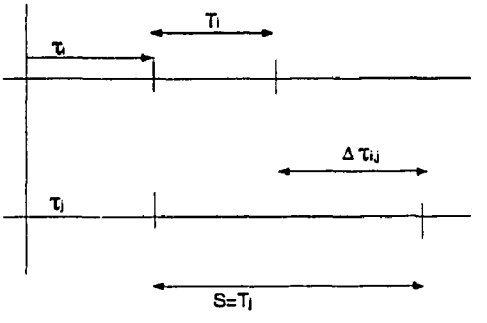

Figure 2.5.4 Illustration of the constraint $S\left|\frac{t_i}{T_i} - \frac{t_j}{T_j}\right| \leq \Delta\tau_{i,j}$ when $\tau_i = \tau_j$.

An illustration of the usefulness of the skew tolerance constraint $S\left|\frac{t_i}{T_i} - \frac{t_j}{T_j}\right| \leq \Delta\tau_{i,j}$ is shown in Figure 2.5.4. In this example, the time average delays for the two data streams i and j are assumed to be equal, *i.e* $\tau_i = \tau_j$. Then, the skew tolerance parameter as derived both by the constraint (see Equation (2.5.15)) and Figure 2.5.4 is shown to be $\Delta\tau_{i,j} = |T_j - T_i|$, and the length of the segment size is $S = \max\{T_i, T_j\} = T_j$. In their synchronization control architecture, the *SQoS* parameters are passed to the *SDS* entity. The *SDS* then undertakes the task of achieving the required quality according to the *SQoS* parameters. This is achieved by determining the correct size of the segment S and the playback times of the packets for each stream in the multimedia segment.

Ravindran and Bansal in [41] propose a synchronization protocol for multimedia data streams similar to [40]. Their paper focuses on maintaining the temporal relations of various data streams in the multimedia information. Maintaining the required association between data units across various streams in real-time requires framing of data streams whereby various points in the data streams deliverable simultaneously to the user are

identified. The transport entity of their protocol enforces synchronization according to quality of transport parameters QOS_{transp}. The QOS_{transp} originates from the application based on inherent temporal relations among various data streams. Those parameters are used by the synchronization protocol in the following two ways:

- map to QOS_{net} parameters at the network interface below to support real-time movement of data segments through the network, and
- generate the correct play back (display) times for the various data streams of the multimedia information segment.

QOS_{net} parameters indicate the average and peak bandwidth required on the channels carrying these data, the acceptable level of delay variation across various channels, and the acceptable end-to-end delay between source and destination. The QOS_{net} parameters influence the allocation of network resources for implementation of the channel.

The overall architecture of the synchronization protocol is defining the list of parameters need to be specified in QOS_{transp}. Then the QOS_{transp} parameters are mapped into the QOS_{net} which define the allocation of the network resources.

Consider an N-channel connection to transport a set of data streams $\bar{X} = \{x_1, x_2, ..., x_N\}$ from one or more sources to one or more destinations, with stream x_r flowing on the r^{th}-channel, $r = 1, 2, ..., N$. The skew tolerance parameters specified by the application at the transport layer interface is given by

$$QOS_{transp} = \{DV, IGS, IFP\}\ , \tag{2.5.16}$$

where *DV* is the divergence vector, *IGS* is the inter-glitch spacing, and *IFP* is the inter-frame pause. The *DV* denotes how far apart the data segments from an interval i can be separated along the temporal axis and be still acceptable to the application. In other words, *DV* specifies a time interval where a *data-slip* is acceptable. Due to frequency mismatches

of the clocks, one data stream x_r which used to belong to the i^{th} synchronization interval (vector $\bar{X}^i = \{x_1^i, x_2^i, ..., x_N^i\}$), might find itself to $\bar{X}^{i+1}$ *or* $\bar{X}^{i-1}$. This phenomenon is called data-slip, and it manifests as the occurrence of a glitch during the presentation of the data frames.

IGS refers to inter-glitch spacing, and denotes the minimum acceptable spacing between two consecutive glitches. For example, in *TV* the human viewer cannot tolerate glitches occurring more frequently than 1 in 300 pictures. The inter-frame pause *(IFP)* denotes the maximum amount of pause that can be tolerated in the flow of real-time when the application advances from the current i^{th} temporal interval to the next $(i+1)^{th}$.

Once the QOS_{transp} are specified for each multimedia synchronization interval i, they are passed to the network layer below. The network layer maps the QOS_{transp} to the QOS_{net} parameters that specify the resource allocation of the network to each channel carrying multimedia information.

Although we won't describe them, we would like to mention two recent works that we consider them important. In [42], Escobar *et.al.* present a flow synchronization protocol, and Znati and Field in [43] present a network level abstraction for multimedia communications.

2.6 Concluding Remarks

An overview of selected papers related to multimedia network synchronization was presented in this chapter. We tried to give the reader a comprehensive view of the multimedia synchronization issue by grouping papers according to the nature of the multimedia synchronization problem they were attacking. An important problem which arises in multimedia synchronization is the phenomenon of network jitter. Due to its

relevance and importance to multimedia synchronization, an entire section (section 2) was devoted for its definition.

In section 2, algorithms for packet voice synchronization were presented. Basically, there are two problems that need to be addressed in packet voice synchronization. Firstly, after determination of the maximum time delay that packets experience, play them back according to this time even if they came earlier. And secondly, determine their correct playback times even if silent intervals exist between them. In section 3, algorithms that determine the proper set of packets (*Fusion Set*), one packet from each participant, that require mixing/playback at the receiver were presented.

Finally in section 4, algorithms/protocols for transporting multimedia objects from multiple sources to multiple destinations were presented. Multimedia synchronization for these protocols refers to two things. Firstly, the temporal relations existed among the various data streams during generation at the sources have to be preserved also during playback at the receiver. Those temporal relations usually have some flexibility described by a set of quality of service parameters (*QoS*). Secondly, coordination of two processes, one existing at the network layer and one at the application is needed. The process at the application layer makes sure that the *QoS* are met, while the process at the network layer makes sure that the network has sufficient resources to allocate for the transmission of the multimedia objects while the *QoS* are satisfied.

Chapter 3 Design, Implementation and Analysis of a Multimedia Conference System Using TCP/UDP Protocols

3.1 Overview

The software we developed is capable of transmitting voice, video, graph points, and data (keyboard characters), for point-to-point multimedia conferences. We can logically view this software as if it were comprised by two parts. One part is the graphical user interface, and the other part is the communication part. The function of the Graphical User Interface part (GUI) is to present the available information to the user-participant. Therefore, the function of the GUI is to display in the screen of the monitor the local and remote images of the two sites, to provide a window where the user can draw pictures that at the same time will also be displayed at the screen of the remote host, view and edit documents that are to be shared with the other participant, and finally, to provide the proper interface for the voice — both for listening and speaking.

The communication part can further be logically divided into two parts: sender and receiver. As the words imply, the function of the sender is to transmit the data to the remote host, while the function of the receiver is to receive it. When a connection is made between two computers, we say that we opened a communication channel, that depending on the protocol used, i.e. either TCP or UDP, we have a TCP or UDP channel respectively. The difference between these two protocols is that one is reliable (TCP) , while the other is unreliable. The software uses both channels. The TCP channel (reliable) is used for transferring data and graph, while the UDP channel (unreliable) is used for any other form of data, i.e. voice or video.

The communication part of the software constructed so that it can work with two channels as follows: voice and video are transmitted through the UDP channel, and data and graph through the TDP. Since the video boards are capable only for 4 video per second, the transmission of the images are not real-time. Also, the packet loss on the UDP channel is less than 1%.

Specifically, our multimedia system is capable of transmitting voice, video, data and graphics between two computers. The capability of our color video board (SUN's videopix) is 4 frames per second. We design the multimedia packet format based on this video frame interval(0.25 seconds). Because the Sun's videopix only provides software JPEG compression, we have not used compression in the consideration of processing overhead. Each video frame is 21600 bytes(120×180 pixels, each pixel of 1byte). Therefore, one multimedia packet holds one video frame plus 0.25 seconds of voice data. Given that 8KHz sampling and 8 bit PCM, the size of voice becomes 2000 bytes since each video frame lasts 0.25 seconds. Two different architectures are implemented and compared. In the first architecture all application services, interactive video, interactive voice and data and graphics, are multiplexed at the application level and transmitted through a TCP channel. In the second architecture interactive video and voice are transmitted through a UDP channel, while data and graphics are transmitted through a separate TCP channel. By graphics we mean data which allows a user to draw using a mouse in a window on the screen. The data represents mainly characters and commands from console.

In this chapter, we present the approaches we have taken in solving specific issues such as the buffering and the interaction schemes between the Graphical User Interface part and the communication part both at the receiving and at the sending ends. In addition, we have developed a new method of estimating the end-to-end delay between

two hosts. The rest of the paper is organized as follows. In the following section, we discuss the multimedia system architecture and the multimedia conference system we have developed. We then describe the multimedia applications such as video, voice and shared workspace processes in section 4. In section 5 we propose and discuss the implementation of two multimedia transport mechanisms. Communications between different processes, including remote process, are described in this section too. In section 6 we present experimental results focussed on the measurement of the end-to-end delay and jitter. In addition, conclusions are drawn from the experiments, and evaluation of TCP/UDP for use as a multimedia protocol is presented.

3.2 General System Architecture

3.2.1 System Configuration

Figure 3.2.1 shows the system configuration for the multimedia network testbed, which consists of an 100 Mbps FDDI as a backbone, 3 servers (Sun workstations 4/330) connected to FDDI, two 10 Mbps Ethernet segments each one connecting several Sun IPC's and PC's. SunOS 4.1.1. is the operating system for Sun workstations and PC's are served by PCNFS. Each of the hosts (Sun workstations IPC) taking part in the multimedia experiment resides on a different Ethernet segment.

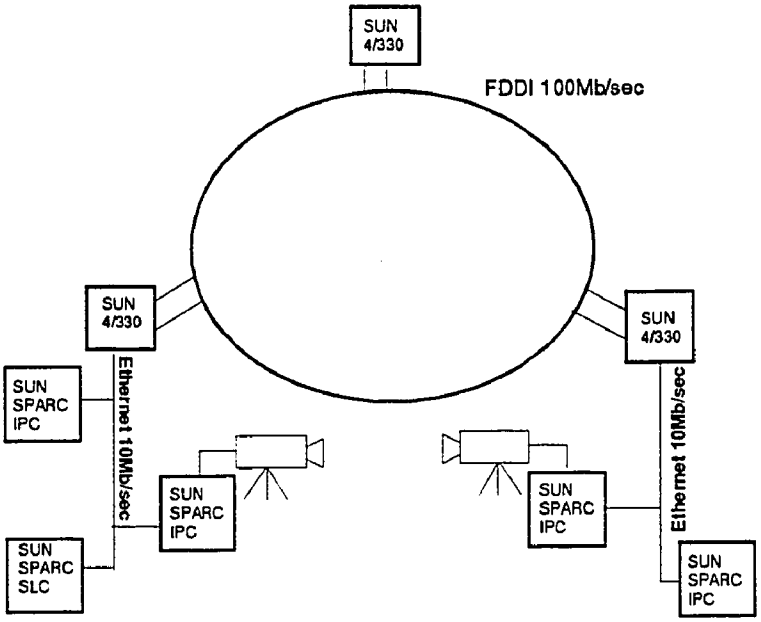

Figure 3.2.1 Testbed Configuration

3.2.2 Multimedia Conference Architecture

The Multimedia Conference System consists logically of three parts: The Multimedia Application, the Sender and the Receiver, as shown in Figure 3.2.2.

Main functions of the Multimedia Application are as follows:

- Graphical User Interface (GUI)
- Interaction with voice and video device drivers and console
- Interaction with Sender and Receiver

The multimedia data from Multimedia Application is passed to the Sender which forwards it to the remote host. The Receiver receives the multimedia data from the Sender at the remote host and then passes it to the Multimedia Application.

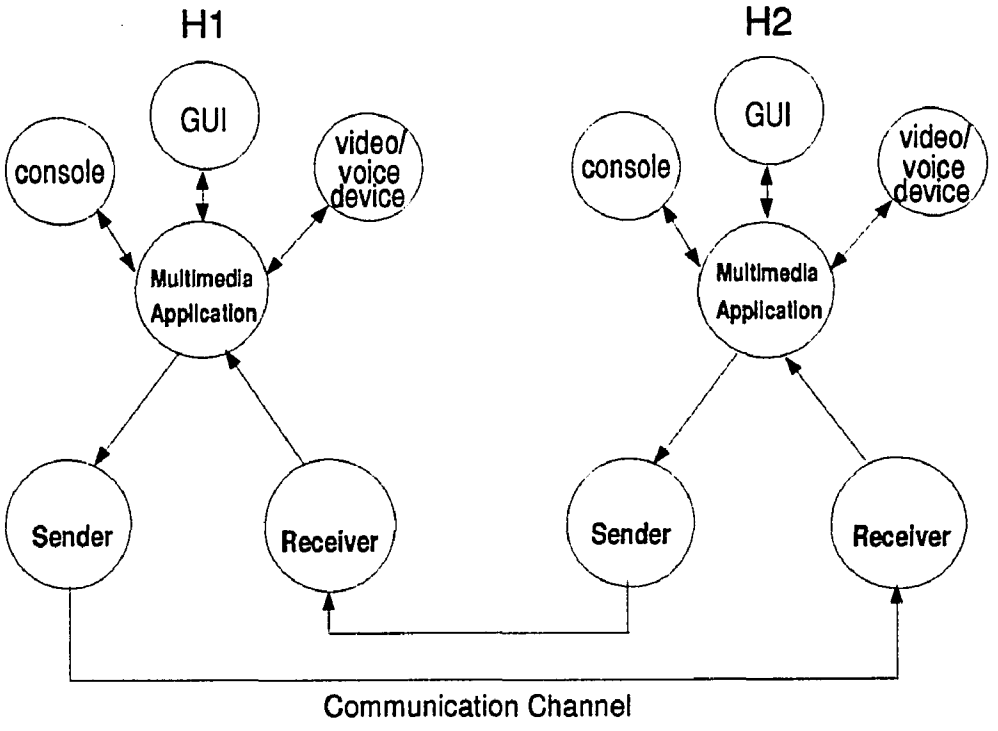

Figure 3.2.2 System Architecture

3.2.3 Multimedia Applications Structure

Multimedia Application includes the graphical user interface and many different processes handling different media. As shown in Figure 3.2.3, these different processes are video process, enlargement video process, voice process, shared workspace process, graphic process and private work process. Below, we discuss graphical user interface and each of the processes.

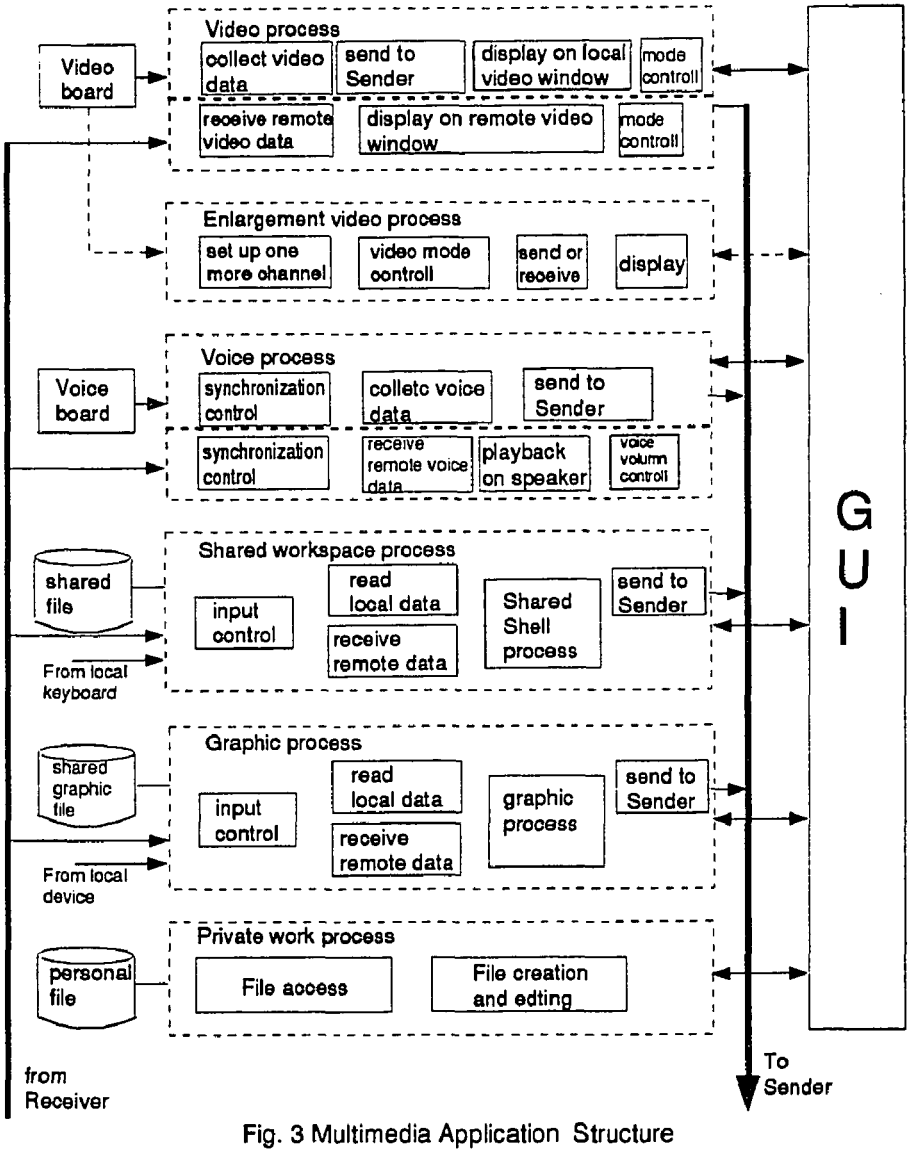

Fig. 3 Multimedia Application Structure

Figure 3.2.3 Multimedia Application Structure

3.3 Design of the Graphical User Interface Part

3.3.1 Graphical User Interface

The Graphical User Interface is written in Xlib, Xview and C. The layout of GUI for multimedia conference appears in Figure 3.3.4. Each user has to press the button "Speak" in the "Voice Controller" window in order to join the conference . In the

local video window there are two more buttons: "Motion" and "Still". The user has the flexibility to send videos to the other participant(s) or one may choose to send just voice by pressing the respective button. In the upper-left corner of the screen there is the button "Enlarge". By pressing this button our local video is magnified by 16 times (the width and the height are magnified by 4) , and this enlarged video is sent to the remote host. This feature is useful when one wants better visual contact. In the upper-left corner there is the "Voice Controller" window which controls the intensity of the voice. Next to the "Voice Controller" are two video windows. The first is the video from the local host and the second from the remote. Below the video windows is the "Shared Workspace" window. This window is shared among the participants of the conference (for the present only two participants are supported). The function of this window is to facilitate simultaneous access to the system by the participants of the conference, and in particular, to enable co-editing of documents. On the upper-right side of the screen is the "Draw Window". The function of the "Draw Window" is to provide the participants with a mean of communicating schematic drawings that are invaluable to the understanding of complex ideas. The "Draw Window" is operated by three buttons: "Erase", "Draw" and "Clear". By pressing the "Erase" and "Draw" button we can either erase objects on the window or draw (the action toggles). By pressing "Clear", we clear the entire window. Finally, on the lower-right side of the screen is the "Private Window". It is used by each participant as his/her private window, and as implied by the name is private to each participant.

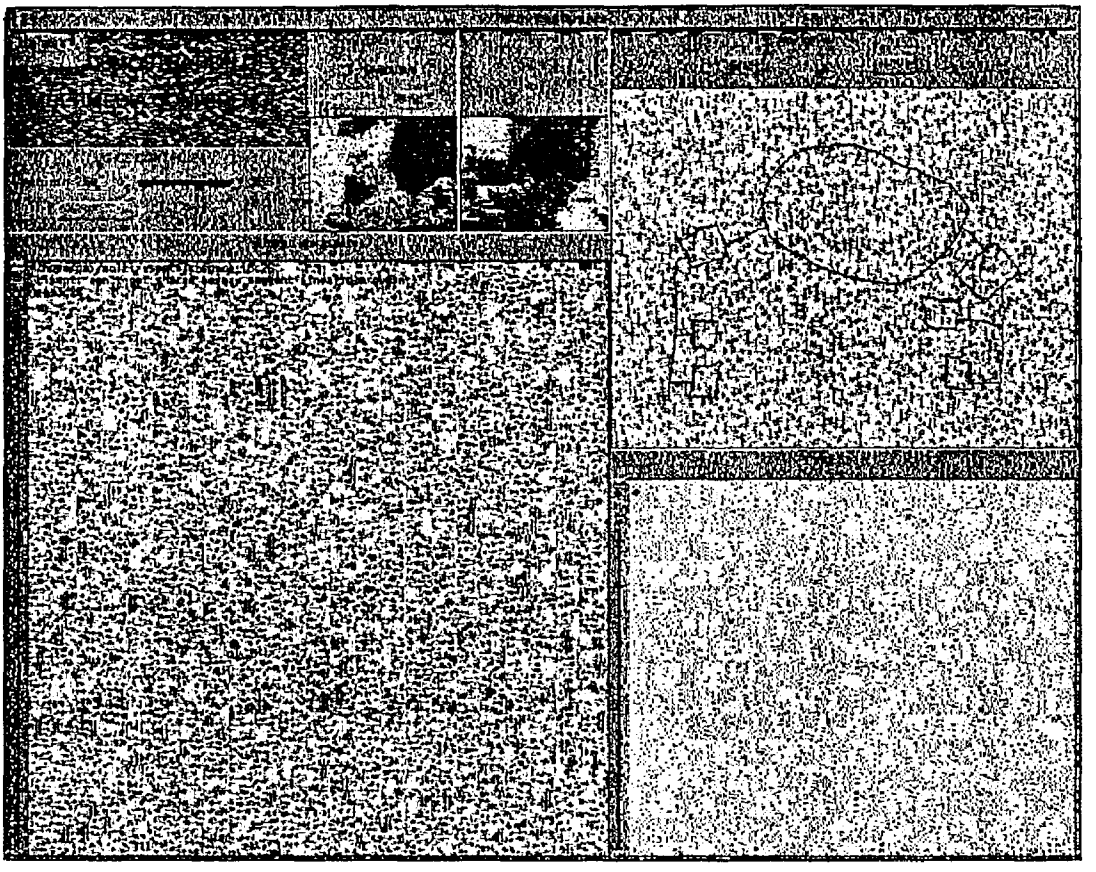

Figure 3.3.4 Graphical User Interface

3.3.2 Processes

Video Process The functions of video process can further be divided into two independent processes: one for handling local interactions (upper part of the video process in Figure 3.2.3) and the other (lower part of the video process) for remote interactions. The process for local interactions includes collecting video data from local video board, sending video data to the Sender and displaying video on the local video window. The process for remote interactions includes receiving remote video data from the Receiver and displaying it on the remote video window. Local video data from local video driver and remote video data from the Receiver are independently and concurrently read and

displayed on local and remote video windows . The video process also controls the video selection mode: voice only, or voice and video.

Enlargement Video Process This process allows video window to be enlarged. When this process is activated, it sets up a separate channel for the transmission of enlarged video data, stops regular video data transmission by changing video selection mode, sends (or receives) the enlarged video data, and finally displays the video data on the screen.

Voice Process The operation of the voice process is similar to the video process except that it handles microphone and speaker instead of video board and video window. Another important function of the voice process is that it is waken up by the Sender to generate synchronization signal and sends it to video process in order the voice and the video data to get synchronized.

Shared Workspace Process This shared workspace process provides a shared space for collaborations among the conference participants. As mentioned above, co-editing of a document represents a good example of such an application. In order to build the shared workspace schell we create a Shared Shell Process. The Shared Shell process accepts commands not only from the local keyboard but also from the remote hosts keyboards through TCP channels. There are two ways to realize the shared workspace by using the Shared Shell: centralized and decentralized. Figure 3.3.5 shows the two methods. In the centralized method there is only one shared shell process. This shell process accepts two inputs, one from the local keyboard and the other from the remote. The shell process outputs the results to local and remote terminals. Two communication channels will be needed for remote host's input and output. In the decentralized method two shared shell processes exist. Each shell process receives two input commands, one from local

keyboard, another from remote shell, and then transmits the results to the local screen only. Two communication channels are needed just for inputs.

Comparison Between the Two Different Methods Let us compare both methods. In general, the trade-offs between the two methods is in processing power, memory space and cost of the channel bandwidth. For instance, consider that a large CAD program is used by many conference participants. If each user has enough memory space and processing power to store and process huge CAD program at his/her local site, it is better to implement decentralized method, since each host only needs to transmit control commands, not the huge data itself. This saves channel bandwidth. On the other hand, if a waste in channel bandwidth is more cost effective than having a powerful host machine, the centralized method may be the proper choice. In addition, it is to be noted that conference management process has to be provided to maintain synchronism among the participants in the decentralized method.

In our multimedia conference system, we have implemented the decentralized method because the time delay is the major concern to support real-time performance. The reasoning behind this is the following. In centralized method the transmission path, which in turn involves transmission delay, for the remote host it takes twice as much when compared to the decentralized method, as it is evident in Figure 3.3.5.

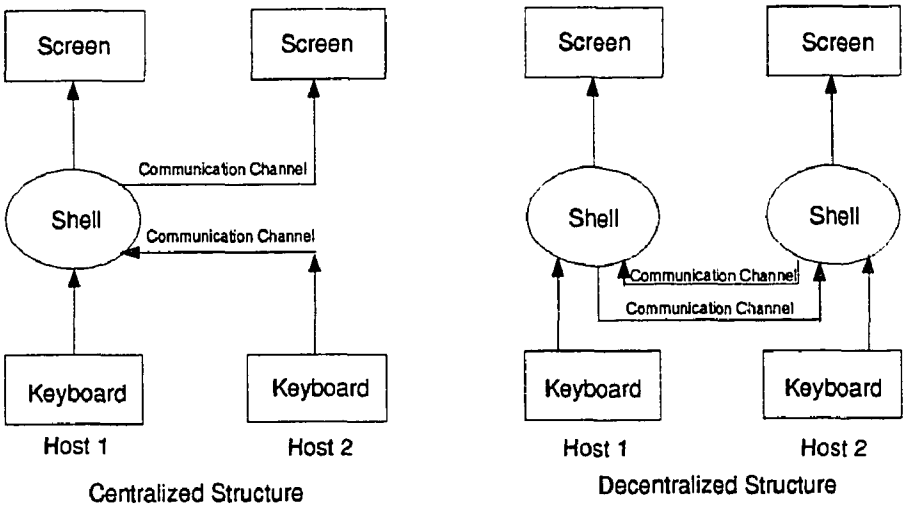

Figure 3.3.5 Structure of Shared Shell

With the decentralized Shared Shell, the Shared Workspace process reads local and remote commands selecting either the local or the remote process and passing them to the Shared Shell. If the command is from the local keyboard, it is sent to the Sender.

Shared Graphic Process This process provides facilities for Mouse to draw, erase and clear. We also use the decentralized method as in the Shared Workspace process. The main functions of shared graphic process are similar to those of the Shared Workspace process. Both graphic processes in host 1 and host 2 are exactly the same. So, if the user at host 1 enters the following command to draw a line

$DrawLine(x1, y1, x2, y2)$,

the graphic process at host 1 will draw a line which begins at (x1, y1) and ends at (x2, y2). At the same time the graphic process at host 2, upon reception of the same drawing command, will draw the same line on the graphic window at host 2. Only graphic commands need to be transmitted through the communication channel rather than the graphic itself. This avoids large graphic data transmission.

Private Work Process This process provides the user with personal document processing capabilities like editing personal text, personal file access, etc.

3.4 Design of the Communications Part

3.4.1 Two Schemes for Multimedia Transport

We send a multimedia message exactly every 0.25 seconds from the main memory to the transport layer even if it had been generated earlier.

We have developed two schemes for multimedia transport. The first, Scheme A, multiplexes all application services, i.e. voice, video, data and graphics, at the application level (or layer) to form a multimedia message and then transmits it through a TCP channel. The advantage of this scenario is that underlying TCP's reliable service is provided to all the application services. In addition, we avoid the synchronization problem among the different media by transmitting all media data in a single message. This scheme A, however, is not without a price: unnecessary processing overhead is required for component services that can tolerate small errors and packet loss. In the multimedia conference environment , different media have different service requirements. For instance, delay sensitive services, e.g. interactive video and voice, may tolerate certain amount of error but not excessive delay, whereas for error-sensitive services, e.g. data and graphics, the opposite is true. One may exploit this characteristic of application services in the protocol design. So the second scheme comes into play.

In the second scheme, Scheme B, we target at reducing the processing time at the transport layer by exploiting the delay sensitive but less error sensitive characteristics of the real-time voice and video services. Thus, we use UDP for interactive video and voice, while TCP is used for data and graphics since they require strict error protection. A handicap with current UDP is its total unreliability making it problematic even for

interactive video and voice. Using LLC type 2 service may alleviate the problem because of its reliable connection-oriented service. As it will be shown by the experiments in our testbed, the quality of voice and video with UDP for 30% Ethernet load is still good.

3.4.2 Multimedia Message Format

The message formats for Scheme A and Scheme B are shown in Figure 3.4.6(a) and 3.4.6(b) respectively. In the message format in scheme A, the first field (1 byte), called VI (Video Identifier), is used to inform the Sender as to whether the current message includes a video field. Sometimes, during a multimedia conference, we may not need motion video but just still image. In this case we can stop transmitting video in order to reduce the traffic on the network since video occupies 90 percent of the multimedia message. The VI serves this purpose.

The second field is the voice field where the voice samples reside. Its size can be either 2000 or 1000 bytes. If we send video, the voice is 2000 bytes otherwise 1000 bytes. The third field (14 bytes) is the timestamp field, which records the time when a message leaves the Sender. This time information is needed to compute the delay and throughput performance measures. The fourth field (2bytes) carries the sequence number associated with each message. We send the sequence number in order to keep track of messages, especially to count the lost messages in the UDP channel. The fifth field holds the video data (a video frame, 21600 bytes). Five more fields follow: 1 byte EI (Enlargement Identifier for video) field which indicates enlarged video data; 1 byte DI (Data Identifier) field which indicates whether there is character data and how long it is; a 5 bytes Data field which holds character data; 1 byte GI (Graphic Identifier) field which indicates if there is graphic data and how many bytes it is; and 5 bytes Graph field which holds the graphic data.

TCP Channel

VI	VOICE DATA	TS	SEQ	VIDEO DATA	EI	DI	DATA	GI	GRAP
1	2000 or 1000	14	2	21600	1	1	5	1	5

(a) Message format of the scheme A

UDP Channel

VI	VOICE DATA	TS	SEQ	VIDEO DATA
1	2000 or 1000	14	2	21600

TCP Channel

EI	DI	DATA	GI	GRAP
1	1	5	1	5

(b) Message format of the scheme B

Figure 3.4.6 Message Formats

In the case of Scheme B, there are two formats for each transport channel. For the UDP channel, the message format consists of the first five fields of scheme A, while for TCP channel it consists of the last five fields of scheme A.

3.4.3 Communications Between Processes

The whole system is implemented in C, XView and XLib under SunOS Unix operating system. The multimedia software consists of three processes as explained in section 4: the Multimedia Application process (MA), the Sender and the Receiver. In this section, we describe the interactions between different processes (local and remote processes): interprocess communications between local processes and inter-host communications. The Unix interprocess communication technique (IPC) is used for the communications between the MA and the processes of the Sender and the Receiver within a host. For communications between different hosts, we use the socket technique.

Consider now the implementation of the two schemes. Scheme A involves only one TCP channel, while Scheme B includes one UDP channel as well as one TCP channel. Following we explain about the implementation for the Receiver and the MA. Implementation between the Sender and the MA is very similar and it is not presented.

Implementation of Scheme A In Scheme A, all multimedia information is multiplexed at the application level and transmitted through a TCP channel. TCP's flow and error control maintain the integrity of the application.

In order to avoid excessive buffer copies between processes, we use the shared memory technique. The problem arising when we use the shared memory is the synchronism among the processes trying to access the shared memory. This is due to the speed disparity existing between the processes of video and voice and processes of the Sender and the Receiver. Semaphore technique solves this problem.

In contrast to the voice and video processes (continuous data streams), data and graphic processes are not always active. At sending host, they are only awaken and send update information upon receiving the input commands. At the receiving host, data and graphic processes are awaken by the Receiver process. Named pipeline technique is well suited for this type of interprocess communications. When the Receiver receives the data or graphic information from the remote host, it activates the appropriate processes and then transmits the data. Note here that for the voice and video data we use the shared memory for transmission due to the large size involved; however, for the data and graphic we use the pipeline to transmit short size data. Figure 3.4.7 shows interactions between the different processes.

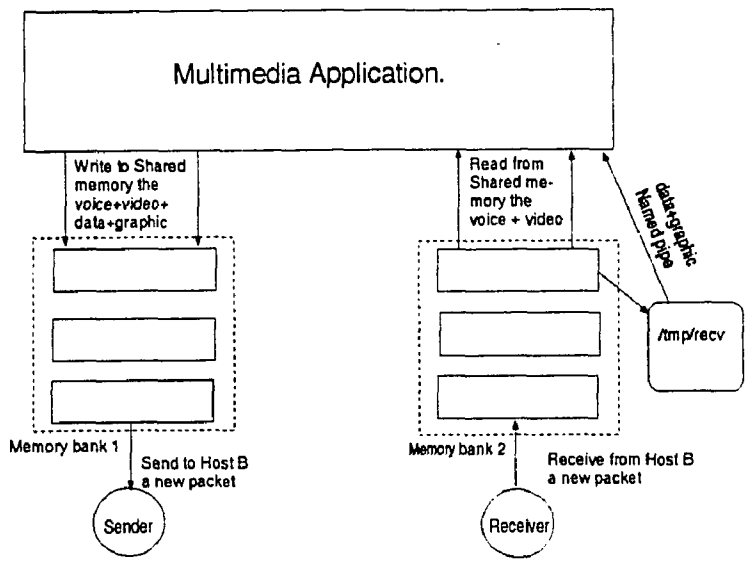

Figure 3.4.7 Interaction between the different processes

We have two banks of shared memory. The bank 1 is used by the MA and the Sender, while bank 2 is used by the MA and the Receiver. Each bank holds three buffers, each buffer holding one multimedia message. Two buffers in each bank are necessary if we want parallel execution for the MA and the Sender/Receiver processes. A third one is needed when the speed of one process does not match the other. More than three buffers in our application seems abusive in memory usage and worsens the performance in terms of average time delay.

We use the Berkeley stream type socket technique to communicate between the two hosts. Since the multimedia message is quite big (about 24Kbytes including 21.6 kbytes video, 2kbytes voice, graphic, data and header information), we use the *writen* and *readn* calls (see [44], p. 279) to make sure that the whole packet is received/sent. Using the system call *setsockopt* we changed the TCP socket buffer size — the RECVBUF and SENBUF options — to accommodate 24Kbytes.

Implementation of Scheme B In Scheme B, video and voice are transmitted through a UDP channel, while data and graphics are transmitted through a TCP channel. It should be noted that in scheme B we have used two separate shared memories: one for UDP channel and the other for the TCP channel. We use a datagram socket for UDP.

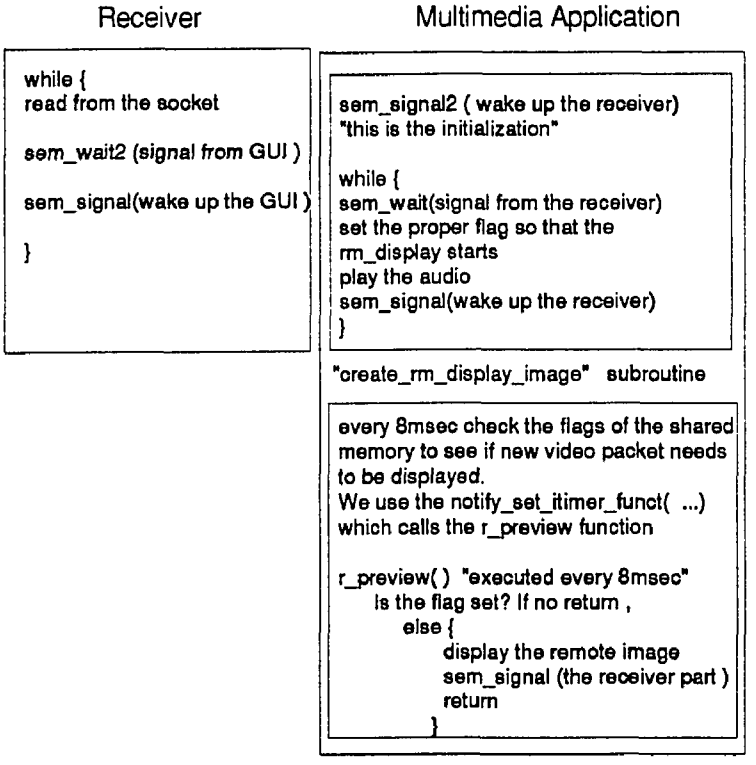

Figure 3.4.8 Communication of the Receiver and the MA processes

UDP Channel One of the major concerns in Scheme B is the packet loss atributed to the UDP channel. In order to minimize this inevitable packet loss, we have implemented concurrent processing technique to reduce the processing time at the receiving host. In general, since the receiver transport protocol is more complex than the sender transport protocol, there should be a means of avoiding receiver overflow problem. In our case, the processing bottleneck exists in the Multimedia Application process. We have compared the performance of the two different implementations: one for serving sequentially for

MA, voice and then video; the other lets MA, voice and video run independently. The latter implementation shows significant improvement over the former in terms of packet loss probability. For instance, in concurrent implementation reading of the periodic voice and video by the Receiver will never be blocked by the slower MA, whereas the sequential implementation does. See Figure 3.4.8 for the exact implementation.

As detailed in the Figure 3.4.8, we use two semaphore functions, sem_wait2() and sem_signal2(), modified from sem_wait() and sem_signal()[11]. Sem_wait2() simply waits for the value of the semaphore in the argument to be decreased by 2 instead of by 1 in order to make the process sleep. Similarly, the sem_signal2() increments the value of the semaphore in the argument by 2 instead by 1. We have done this because in the MA we actually take two actions when we receive messages; playbacks of remote video and voice. When each one of these actions terminates, it wakes up the Receiver by using sem_signal(). Both of the two actions have to be completed in order to wake up the Receiver. Therefore, we employ sem_wait2() at the Receiver.

TCP Channel As mentioned above, TCP channel has its own shared memory. In contrast to the shared memory in UDP, only one buffer is allocated in the TCP shared memory. The reason is as follows. When one transmits data and graphic information from keyboard or mouse to the Sender, the processing speed of the Sender is faster than the data and graphic input rates. In turn, the incoming rates of data and graphic information to the Receiver at the remote host is also limited by typing and moving speed of mouse. Thus, the data and graphic displaying speed is faster than the Receiver process. Since the amount of information in the TCP channel is small, we set the option TCP_NODELAY (see [44], p. 314). This tells TCP to send small messages as soon as it receives them.

3.5 A New Method for Finding the Time Delay and Experimental Results

3.5.1 Clock Offset Estimation

Accuraty of the measurements between two different machines has always been the interesting subject in the area of computer communications due to the absence of a global clock [21]. It is not enough for someone to find the time offset between two machines but he also needs the frequency offset, i.e. the difference in the rates between the two clocks, in order to make truthful measurements for the entire duration of the experiment.

In the case of a multimedia conference system we have ascertained a way to avoid the frequency offset problem by continuously updating the time offset. It is required that both ends send and receive the same amount of messages at the same time. As is illustrated in Figure 3.5.10, we send 1 message every 0.25 seconds. When we send a message from host A to host B, we time stamp it with time T_a. At the receiving end (host B) we time stamp it after network delay T_d with time T_b. The end-to-end delay time that appears at host B is $T_b - T_a$. We denote this time as T_{a-b} . This time also equals to the real end-to-end delay time T_d plus the time offset between host A and host B . In other words, $T_d = T_{a-b} - T_{off}$.

At about the same time, host B sends a packet to host A, and the end-to-end delay for this message equals to $T_d = T_{b-a} + T_{off}$. If we add these two equations we get $T_d = (T_{a-b} + T_{b-a})/2$. This process of estimating the delay time is an averaging process since it is repeated for all the messages sent during the experiment.

There is an assumption that may cause an error, however small. Due to small variations in the processing speed of the two computers and random variations of the traffic in the network, the number of messages received by each computers will not be the same. This discrepancy however can be measured and can be taken into consideration

if it is large, otherwise it will be ignored. In our case we have observed that for a total of 2000 messages one computer receives, the other receives 2000 ± 20 messages. This means that the 2000th message will be compared with the 1980th for example, and that will give us an error equal to the time offset caused by the difference in the rates of the two clocks in the time period of 5 seconds (20 packets times 0.25 seconds per packet = 5 seconds). Knowing that in our two computers the frequency offset causes a time offset of about 1 second every half an hour, the error is negligible.

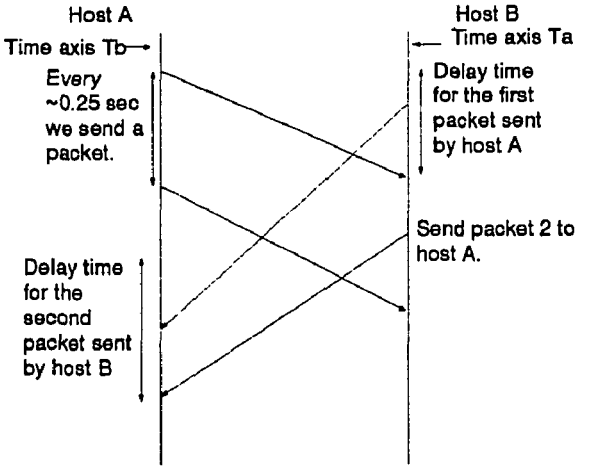

Figure 3.5.9 Timing Diagram between the two computers to estimate the average delay

3.5.2 Experiments and Analysis of the Results

Figure 3.5.10 shows the histograms of typical runs of 5000 multimedia messages for both Scheme A and Scheme B under 2 conditions: zero traffic and 30% background traffic utilization. In this experiment the multimedia message only contains the voice and video data. Because each multimedia message is sent every 0.25 seconds, the network jitter can be thought as the standard deviation of the delay. Thus, the end-to-end delay performance of Scheme A is measured at the TCP channel and for Scheme B it is measured at UDP channel. In the case of Scheme A with 0% background traffic, the

average end-to-end delay is 98 msec and the network jitter is 38msec. When we increase the background traffic to 30%, the average delay increases to 146msec and the jitter to 70msec. In Scheme B with 0% background traffic the average delay is 46.3msec and the jitter is 15.6msec. For this particular run 35 messages have been lost in 5000 messages. When we increase the background traffic to 30%, the average delay increases to 58.5msec and the jitter to 22msec. The message loss goes to 100 per 5000 messages. The packet loss probability varies, but it never goes more than 1% for 0 % background utilization and more than 2.5% for 30% background utilization. In both background utilizations Scheme B (UDP) seems to perform better in supporting real-time multimedia services.

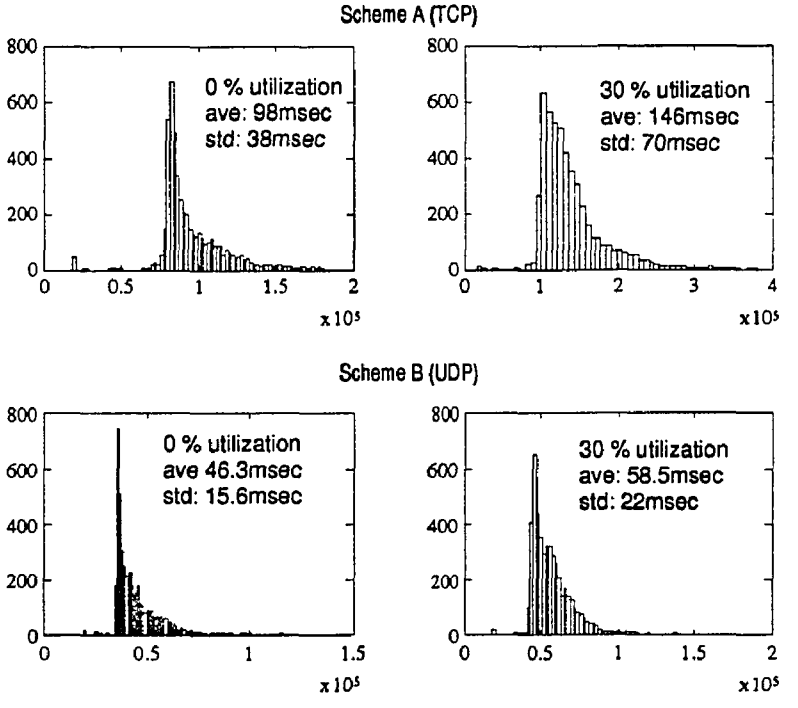

Figure 3.5.10 Measurements of End-to-End Delay for Scheme A and Scheme B

With the 4 frames per second video capability, it is hard to evaluate the video quality. Instead, we subjectively tested the voice quality. The UDP implementation seems better than the TCP implementation in voice quality thanks to the shorter delay.

How many conferees can be supported by this system? Since every multimedia message is 24Kbytes, the total traffic generated at the application level due to a point-to-point multimedia conference application is 24Kbytes x 4 messages/sec x 8bits/byte x 2 (bidirectional traffic) = 1.6Mbits/second. Assume that the traffic at the physical layer is about twice as much[13], say 3.2Mbits/sec. Since the Ethernet is capable of 10Mbits/sec, it can sustain up to two such multimedia conference applications. If the network is heavily loaded, then only one such conference application can be sustained.

Chapter 4 Point-to-Point Single-Medium Synchronization in the Presence of Noncontinuous Periodic Traffic in Real-Time Network Applications.

4.1 Overview

Herein the problem of point-to-point synchronization in the presence of noncontinuous periodic traffic in real-time applications is discussed. Our present work work constitutes the first —and the most fundamental — building block of our proposed Multimedia Synchronization Protocol. Our protocol consists of two more blocks: one block is related to intermedia synchronization (see chapter 5 or [45, 46]) and the other is related to interparticipant synchronization (see chapter 6 or [33, 32]). The model of the traffic is noncontinuous periodic which nicely suits the voice traffic with silent-talkspurt intervals. Factors influencing synchronization are network jitter or variability of the delay and frequency offset between the clocks at the sender and at the receiver. The definition of the network jitter and its estimation is carried out using the concept of the reference times that we will introduce in this chapter. In addition, a scheme that permits the receiver to uniquely identify the sequence number of each packet using a minimum number of bits and its proper implementation are presented. The optimal number of bits for a point-to-point communication considering only a single medium is found by exploiting the fact that the maximum time duration the numbering scheme should span needs not to be greater than the maximum variation of the jitter expected in the network.

For the case of point-to-point single medium (either video or voice) communication, p_1-bits from the control field of each packet will be devoted for synchronization. These synchronization bits are necessary in order for the receiver to determine at all times

the playback time of each packet. The proposed sequence numbering scheme for synchronization is shown in [45, 46] and in chapter 5 to be sufficient to the case of intermedia synchronization. The number of bits are now called p_ψ, where $p_\psi \geq p_1$. An added functionality of these p_ψ—bits is that these bits are also used to identify the difference of the average time delays for the two media. The number of bits needed for sequence numbering is proven to be relatively small (2 to 4 bits). This is an important result especially for the future *ATM* networks, where due to the small and fixed length of the packet, minimization of the control bits is a pressing need [26, 47]. A synchronization algorithm which does not use any interchange of control messages can be found in [12, 13, 14].

In section 2, the background definitions of the network jitter and of the reference times introduced in [33, 32] are repeated here for convenience and completeness. In section 3, the estimation of the reference times in the presence of noncontinuous periodic traffic is presented. Subsequently, in section 4 the bounds of the network jitter and then the minimum waiting time the receiver has to wait for a packet to arrive are determined. In section 5, our algorithm for determining the minimum number of bits for sequence numbering in the case of point-to-point single medium communication is presented. Finally, in section 6, we present our concluding remarks.

4.2 Modeling the Traffic as Noncontinuous Periodic.

In Figure 4.2.1, a graphical representation of a voice traffic being modeled as noncontinuous periodic is displayed. The shaded regions represent talkspurt intervals. Voice packets are generated at the sender exactly at the points indicated in the graph with the vertical lines. This means that a voice packet which is at the front or at the end of a talkspurt interval may only be partially filled with voice data. This modeling of

the voice traffic wastes on the average one voice packet per talkspurt interval, but it has many advantages related to multimedia synchronization that will become obvious as we go along. In addition, a sequence number representing the sequence number of the time period the packet was generated at the sender will be attached to the packet. As a result, the packet generated at the time period (slot) between the vertical lines 2 and 3 will be assigned the sequence number 2, the second the number 3, and the sixth the number 8 since a silent interval of one period intervenes.

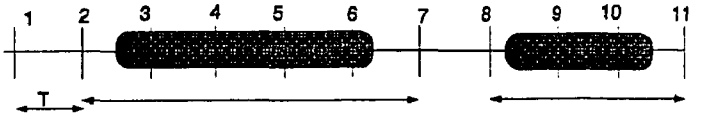

Figure 4.2.1 Representing the traffic as noncontinuous periodic.

The video traffic can also be viewed as *periodic* when the requirement that a video frame (packet) will only be generated at the beginning of a period holds. The validity in modeling the video traffic as *noncontinuous* is best illustrated with an example. Let us suppose that a multimedia conference among five participants is held through a computer network. A participant may listen to all other participants at the same time because all the voice packets will be added together and then played back. In the case of video, however, it will be unwise to look at four different people at the same time. This is not only a matter of taste, but also a matter of space on the computer screen. In addition, the cost of a multimedia conference will be high if exchange of video data occurs in all of the connections of the participants in a conference. Thus, one might choose to display on the screen by default only the person currently talking and by choice anyone else. But by turning on and off a particular participant results in a noncontinuous traffic for the video too.

4.3 Definitions of the Network Jitter and Reference Times.

As shown in Figure 4.3.2, the sender periodically transmits packets with *nominal* period T. However, due to frequency mismatches between the clock at the sender and the clock at the receiver, the actual transmission period of the sender as seen with the receiver's clock will be T^s. The average delay time packets experience to arrive at the receiver from the sender is denoted as D. If there were no variation in the delay and each packet were experiencing a constant delay D, then packets would be arriving at points indicated in the graph as $t_{ref,n}$. These points are called the reference times and they satisfy the following condition

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} (t_n - t_{ref,n}) \to 0 \quad . \tag{4.3.1}$$

If the exact location of the arrival time of one packet, let us say packet n_1, which experienced exactly the average delay D, could be found, then the reference time of any other packet n_2 is given by the following expression

$$t_{ref,n_2} = t_{ref,n_1} + (n_2 - n_1)T^s \; . \tag{4.3.2}$$

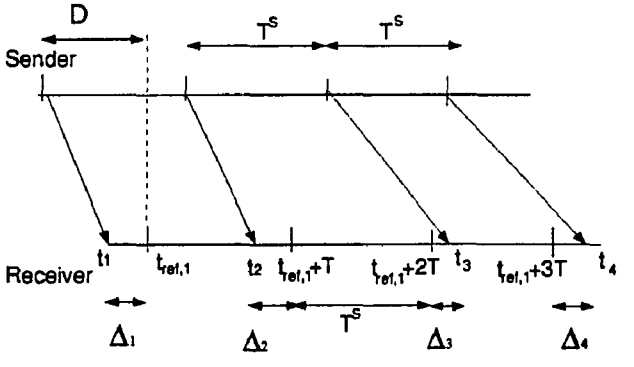

Figure 4.3.2 The definition of the network jitter.

If the actual packet arrivals are denoted as t_n, then the network jitter each packet experiences is defined as the difference of the actual arrival time minus the corresponding reference time

$$\Delta_n = t_n - t_{ref,n} \,. \tag{4.3.3}$$

Different definitions of the network jitter can be found in [3, 29]. Elaborate definitions of jitter, burstiness and the relationships among them can be found in [27, 48, 49, 28, 50].

4.4 Reference Time Estimation

In our work presented in [33, 32], a solution to the multiparticipant multimedia conference problem was presented under the assumption that packets are generated continuously and periodically at the sources. In this section, the determination of the reference times for each arriving packet is investigated in the presence of silence-talkspurt intervals.

The time duration of one frame (for video) or of one voice packet will be designated as T. The periods for different media are assumed to be different, but for the moment T will be either the generation period of the video frames or of the voice packets but not *both*. The time T will be the *nominal* time of the generation period, and when a need arises to indicate the actual periods of the clocks at the sender or at the receiver, the superscripts s or R will be used. Thus, T^s will mean the generation period of the packets (frames) measured according to the universal clock when the sender thinks its generation period is T. Because our approach is based on statistically evaluating the "*reference times*", during the connection-setup time packets will be transmitted continuously and periodically from the sender to the receiver in order a first "rough" estimation of the reference times to be achieved. Exact performance evaluation in calculating the reference

times and design trade-offs among confidence interval, probability that the estimation is outside the confidence interval, and number of packets received, is done in [33, 32].

In order to calculate the reference time (for more details see [33, 32] or chapter 5 in this Thesis), the arrival time t_n of each packet is registered. Assuming N packets have arrived at the receiver and using Equations (4.3.2) and (4.3.3), one can express the arrival times t_n in terms of the jitter that each packet experiences and the reference time of the $\lceil \frac{N}{2} \rceil$'th packet[1] as follows

$$t_n = t_{ref,n} + \Delta_n = t_{ref,\lceil \frac{N}{2} \rceil} + \left(n - \left\lceil \frac{N}{2} \right\rceil \right) T^s + \Delta_n \ , \ 1 \leq n \leq N. \qquad (4.4.4)$$

Then the jitter $\Delta_{\lceil \frac{N}{2} \rceil}$ the $\lceil \frac{N}{2} \rceil$'th packet experiences passing through the network can be estimated as shown below

$$\widehat{\Delta}_{\lceil \frac{N}{2} \rceil} = \frac{1}{N} \sum_{n=1}^{N} \left\{ t_{\lceil \frac{N}{2} \rceil} + \left(n - \left\lceil \frac{N}{2} \right\rceil \right) T^R - t_n \right\} . \qquad (4.4.5)$$

The preceding calculation of the reference time of each packet is based on the fact that packets are generated continuously and periodically; however, in an actual application such as voice conversation packets although will exhibit a periodicity, nevertheless they will not be generated at every period. More specifically, there are going to be silence-talkspurt time intervals, and packets will be periodically generated only at the talkspurt intervals. For this case, a modification to Equation (4.4.5) is needed. Supposedly out of a time interval *(N+k)T* seconds, the aggregated talkspurt time interval is *NT* seconds. As a result, only N packets are sent-received in $N+k$ time slots. To estimate the reference time of each packet in the presence of silence-talkspurt intervals, first the time X with the property that the contributions of the time offsets of all N packets in the interval *(N+k)T*

[1] $\lceil \ \rceil$ is the notation of the ceiling, i.e. the smallest integer greater or equal of the argument.

cancel out ought to be found. This leads to the following equation

$$\sum_{(n\in\mathfrak{N}_E)} (X - t_n) = \sum_{(n\in\mathfrak{N}_L)} (t_n - X)\,,$$
$$(n \in \mathfrak{N}_E : \forall n \in \mathfrak{N}_E \Rightarrow t_n \leq X) \qquad (4.4.6)$$
where
$$(n \in \mathfrak{N}_L : \forall n \in \mathfrak{N}_L \Rightarrow t_n > X) \quad .$$

In other words, $\mathfrak{N}_E$ is the set of packets with the property that for every packet n belonging to this set, the arrival time t_n of this packet is less than time X. Similarly for set $\mathfrak{N}_L$; a packet n will belong to set $\mathfrak{N}_L$ if its arrival time t_n is greater than time X. Notice also that if the number of the elements in set $\mathfrak{N}_E$ is N_E and the number of the elements of the set $\mathfrak{N}_L$ is N_L, then their sum equals the total number of the received packets $N = N_E + N_L$. The set $\mathfrak{N}$ is a proper subset of the set of integers from 1 to $N+k$, $\mathfrak{N} \subseteq \{1, 2, ..., N + k\}$, and $\mathfrak{N} = \{1, 2, ..., N\}$ iff $k = 0$. When $k = 0$, this means that there is not an empty time slot, and there have arrived as many packets as time slots of duration T.

Although N_E, N_L and X are all unknown quantities, Equation (4.4.6) can be solved for X by arranging N_E and N_L to appear with their known sum N. In equation form this leads to the following

$$(N_E + N_L)X = \sum_{(n\in\mathfrak{N}_E\cup\mathfrak{N}_L)} t_n \Rightarrow X = \frac{1}{N}\sum_{(n\in\mathfrak{N})} t_n\,,\ \mathfrak{N} = \mathfrak{N}_E \cup \mathfrak{N}_L \qquad (4.4.7)$$

The next step is finding which packet arrival time t_n exhibits the minimum —in absolute value — distance to time X, i.e. $|X - t_x| = \min_n \{|X - t_n|\}$, $n \in \mathfrak{N}$. If we name the arrival time for this packet t_x, where $x \in [1, ..., N + k]$, then the network jitter this packet has experienced will equal to

$$\widehat{\Delta}_x = \frac{1}{N}\sum_{(n\in\mathfrak{N})} \left\{t_x + (n - x)T^R - t_n\right\} - \mathcal{E}\,, \qquad (4.4.8)$$

where $\mathcal{E}$ is the unknown time offset acquired in time $(X - t_x)$. Exact representation of the error due to the time offset acquired in the time distance $(X - t_x)$ is presented in Appendix A . The validity of Equation (4.4.8) may be questioned on grounds that the uncertainty caused by the error term can be prohibitively high. This can be true in a situation where around the time X a big silence interval exists, and therefore the distance of the nearest packet arrival time to time X is large. However, as it will be proved shortly, this makes no difference for the estimation of the reference times. The error attributable to the frequency offsets of the clocks in estimating the reference times will be of the same magnitude as if the times X and t_x coincided. To see that, an approximation to the sequence number of the reference interval n_X the time X falls into is needed to be found. Graphical representations of X, n_X, t_x, x, and $t_{ref,x}$ are shown in Figure 4.4.3. We assume that a rough estimation of the reference time $t_{ref,0}$ is already available to the receiver during the connection-setup time. $t_{ref,0}$ will indicate the reference time of the time interval exactly one period before the first $N+k$ time intervals considered. Then n_X can be approximated as follows

$$\left(n_X - \left\lceil \frac{\Delta_{min}}{T^R} \right\rceil\right) T^R < X < \left(n_X + \left\lceil \frac{\Delta_{max}}{T^R} \right\rceil\right) T^R \Rightarrow n_X \approx \left\lfloor \frac{X}{T^R} \right\rfloor , \qquad (4.4.9)$$

since $X \gg \Delta_{min}$ and $X \gg \Delta_{max}$.

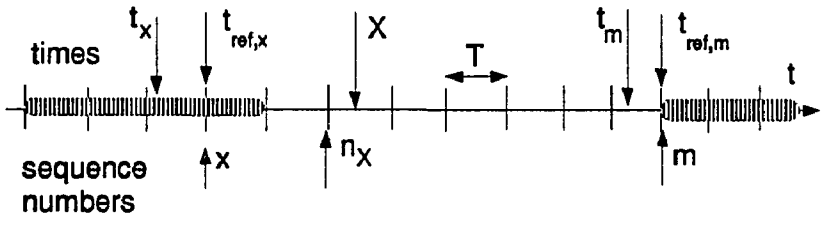

Figure 4.4.3 Graphical representations of X, n_X, t_x, x, and $t_{ref,x}$.

The error $\mathcal{E}$ in Equation (4.4.8) which was derived in Equation (9) of Appendix A can now be expressed in the convenient form

$$(X - t_x) \times \frac{(T^R - T^s)}{T^R} \approx -(n_X - x) \times \left(T^s - T^R\right) . \qquad (4.4.10)$$

With Δ_x available and Equations (4.3.2) and (4.3.3), the reference times for the other packets are obtained as

$$\widehat{t}_{ref,n} = t_x - \widehat{\Delta}_x + (n-x)\left[T^R + \left(T^s - T^R\right)\right] - (n_X - x)\left(T^s - T^R\right)$$

$$\Rightarrow \widehat{t}_{ref,n} = t_x - \widehat{\Delta}_x + (n-x)T^R + (n - n_X)\left(T^s - T^R\right) \ , \ n, x \in \mathfrak{N} \ , n_X \notin \mathfrak{N} \qquad (4.4.11)$$

where an additional error of the estimation of the reference time for any other packet n in Equation (4.4.11) comes from the fact that the term $\left(T^s - T^R\right)$ is unknown. However as shown in Equation (4.4.11), the coefficient of the unknown rate of the time offset $\left(T^s - T^R\right)$ is the same coefficient as if the sequence numbers x and n_X were the same. This ends the prove of the assertion made earlier that the existence of a large distance $|X - t_x|$ will produce no further error in the estimation of the reference times.

4.5 Bounds on the Network Jitter and Minimum Waiting Time

The maximum and minimum bounds of the network jitter are estimated as follows

$$\widehat{\Delta}_{max} \leq \max_{n \in \mathfrak{N}} \left\{t_n - \widehat{t}_{ref,n}\right\} + \max\left\{n_X, N + k - n_X\right\} \times 10^{-\nu} T^R \ , \qquad (4.5.12)$$

and

$$\widehat{\Delta}_{min} \geq \min_{n \in \mathfrak{N}} \left\{t_n - \widehat{t}_{ref,n}\right\} - \max\left\{n_X, N + k - n_X\right\} \times 10^{-\nu} T^R \ . \qquad (4.5.13)$$

It should be noted that we haven't presented yet a method that maps the packet arrival times t_n with their corresponding reference times $t_{ref,n}$. Actually, so far we don't even

know the correct mapping between t_n and n, i.e. although the packet arrival times t_n are known , n are unknown. The correct mapping among the parameters $(n, t_n, t_{ref,n})$ will be shown in the next section (see Equations (4.6.2.19) and (4.6.2.20)).

The second terms in Equations (4.5.12) and (4.5.13) are due to the unknown error $(n - n_X)(T^s - T^R)$ which appears in Equation (4.4.11). Since the minimum or maximum jitter can occur in any of the received packets, we can bound the error by considering the worst case situation which is $\max\{n_X, N + k - n_X\}$. In the same way we can bound the second factor of the second term, i.e. the factor $|T^s - T^R|$ by assuming that the maximum rate of the time offset that anybody can observe in the network cannot exceed a certain value, i.e. $|T^s - T^R| \leq 10^{-\nu} \times T^R$, where ν is a small number (3 to 6).

For point-to-point communication considering only one medium, the synchronization function at the receiver should simply be the inverse of the maximum delay. Packets should wait exactly so much time, as much as it is estimated that all packets will arrive within this waiting time interval. If a packet does not come within this interval, then the packet will be considered lost. Therefore, the waiting time for point-to-point network communication will be

$$W = \widehat{\Delta}_{max} + \epsilon + (N + k) \times 10^{-\nu} T^R = \max_{n \in \mathfrak{N}} \{t_n - \widehat{t}_{ref,n}\} + \epsilon + [\max\{n_X, N + k - n_X\} + (N + k)] \times 10^{-\nu} T^R . \quad (4.5.14)$$

The error $\epsilon = -\frac{1}{N} \sum_{n \in \mathfrak{N}} \Delta_n$ is due to the finite N considered in estimating the jitter of the x^{th} packet. The term $(N + k) \times 10^{-\nu} T^R$ is an upper bound in the error caused by the unknown time offset $|T^s - T^R|$ in the next $(N + k) \times T^R$ time interval, which will be the time that the algorithm will be re-executed. As noted previously, the rate of the time offset $\frac{(T^s - T^R)}{T^R}$ is assumed to be bounded by $10^{-\nu}$, i.e. $|T^s - T^R| \leq 10^{-\nu} T^R$. The error terms ϵ and $[\max\{n_X, N + k - n_X\} + (N + k)] \times 10^{-\nu} T^R$ define the designer's

tolerance in accepting a waiting time that is so much greater from the optimal waiting time Δ_{max}. Subsequently, given a specified tolerance the frequency of the execution of the synchronization algorithm, i.e. the time interval $(N + k) \times T^R$, can be determined.

4.6 Sequence Numbering Scheme for Point-to-Point Single-Medium Synchronization

4.6.1 Determination of the Optimal Number of Bits for Sequence Numbering Identification

To achieve synchronization under all circumstances such as possible out of sequence arrival of a packet, a proper sequence numbering scheme is needed. This scheme should be realized with the minimum number of bits if loss of bandwidth due to the transmission of a large number of control bits per packet is not tolerable. Although both video and voice will exhibit noncontinuous periodic pattern of their traffic, in this section we will mostly speak about voice since bandwidth considerations will arise mostly (or in all times) for the voice packets.

In this section, a thorough evaluation of the minimum number of bits needed to represent a time duration equal to the maximum jitter expected in the network is carried out. An implementation of the synchronization scheme using the minimum number of bits determined here is presented in the following section. The sequence numbering scheme for point-to-point single medium communication using the minimum number of bits is given by the following theorem.

Theorem 1*The sequence numbering scheme for point-to-point single medium communication that uses the minimum number of bits and uniquely identifies the sequence number for each of the arriving packets needs to span a time duration not greater than the maximum jitter* $J = (\Delta_{max})_g - (\Delta_{min})_g$ *expected in a given network.*

Proof: Let us denote with p_1 the number of bits of the sequence numbering scheme. We will denote by t_n the arrival time of the packet that its sequence number is needed to be recognized at the receiver, and as t_m , $m \in \mathfrak{M}$ the arrival times of the voice packets that have identical p_1 -bit sequence numbers, i.e. $(t_n)_{p_1} = (t_m)_{p_1}$ but $m \neq n$. This in effect means that the elements of the set $\mathfrak{M}$ are generated as follows $m = n \pm k2^{p_1}$, $k = 1, 2, ...$ It suffices to distinguish the $n^{'th}$ packet from any other packet with identical p_1-bit sequence number if it is agreed that every time $g_n - g_m > 0$, then $t_n - t_m > 0$, or every time $g_n - g_m < 0$, then $t_n - t_m < 0$.

It is clear that if the conditions presented above are satisfied for one immediate neighbor of n, i.e. $m = n \pm 2^{p_1}$, then it will hold for all other $m \in \mathfrak{M}$. Let us have $m = n + 2^{p_1}$. Then, $g_n - g_m = g_n - (g_n + 2^{p_1}T) = -2^{p_1}T < 0$. Therefore, we have to prove that $t_n - t_m < 0$ whenever $2^{p_1}T > (\Delta_{max})_g - (\Delta_{min})_g$. We have:

$$t_n - t_m = t_{ref,n} - t_{ref,m} + \Delta_n - \Delta_m = t_{ref,n} - (t_{ref,n} + 2^{p_1}T) + \Delta_n - \Delta_m > 0 \Rightarrow$$

$$2^{p_1}T > \max\{\Delta_n - \Delta_m\} = \max\{\Delta_n\} - \min\{\Delta_m\} = (\Delta_{max})_g - (\Delta_{min})_g \quad (4.6.1.15)$$

In Equation (4.6.1.15) we have used $t_{ref,m} = t_{ref,n} + 2^{p_1}T$ since $g_n - g_m = -2^{p_1}T$ and $t_{ref,n} = g_n + D$, $t_{ref,m} = g_m + D$. □

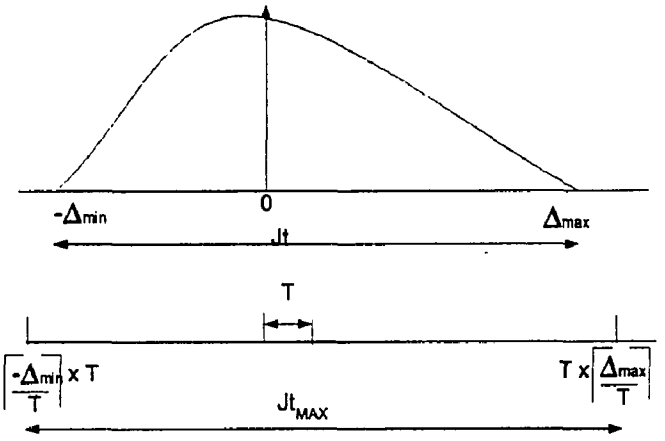

Figure 4.6.4 Graphical representation of the time a proper sequence numbering scheme should span.

H. Schulzrinne [51] describes all the encoding standards used today for the transfer of voice through the internet. The important parameters for determining the optimum number of bits p are the sampling frequency f_s and the number of quantization bits q used. As is described in [51], the sampling frequency f_s can take values from the set 8, 11.2, 16, 22.05, 44.1 and 48 KHz, and the number of quantization bits q is 8 or 16 bits/sample. The time duration $T_{\texttt{packet}}$ one packet lasts when played back at the receiver can be computed as a function of the sampling frequency f_s, the length of the user portion of the packet in bytes l, and the quantization number of bits q used, as follows

$$T_{\texttt{packet}} = \frac{l \times 8 \frac{\texttt{bits}}{\texttt{packet}}}{f_s \frac{\texttt{samples}}{\texttt{sec}} \times q \frac{\texttt{bits}}{\texttt{sample}}} \quad . \tag{4.6.1.16}$$

Two cases will be distinguished. In the first case, the voice packet will be comprised by simply one *ATM* cell while on the second case, the voice packet will be comprised by multiple number of cells c. The reference made to *ATM* cells does not imply that the following analysis will only be valid for pure ATM networks. Any packet route transversing different type of networks, such that ethernet or *FDDI* to *ATM* to ethernet or *FDDI* again, at one point the packet will have to be fragmented into *ATM* cells

passing though an *ATM* network. Since the smallest fixed length packet a packet can be fragmented to is met in the *ATM* network, the results derived in this section will be general enough to include cases of both pure *ATM* networks and hybrid ones. As shown in Figure 4.6.4, suppose that the maximum jitter J expected from the network is $J = (\Delta_{max})_g - (\Delta_{min})_g$, where $(\Delta_{min})_g$ and $(\Delta_{max})_g$ are the maximum values of the bounds of the network jitter that can be observed in the network. Then, for the case of a single cell per voice packet, the optimum number of bits p needed to properly represent the sequence number for each cell-packet is

$$\begin{aligned} J_M &\equiv \left(\left\lceil \frac{(\Delta_{max})_g}{T_{\text{packet}}} \right\rceil - \left\lceil \frac{(\Delta_{min})_g}{T_{\text{packet}}} \right\rceil \right) \times T_{\text{packet}} \leq 2^{p_1} T_{\text{packet}} \\ &\Rightarrow \; p_1 = \left\lceil \log_2 \frac{J_M}{T_{\text{packet}}} \right\rceil = p \;,\; T_{\text{packet}} = T_{\text{cell}} \quad . \end{aligned} \tag{4.6.1.17}$$

In Equation (4.6.1.17) J_M is used instead of J because we wanted to make sure that by placing the average point of the packet arrivals to a beginning of an interval T, there are still enough intervals T to cover both the maximum and minimum values of the network jitter (see Figure 4.6.4). For the case of multiple number of cells c per voice packet, the number of bits needed is

$$\begin{aligned} &J_M \leq 2^{p_1} T_{\text{packet}} \Rightarrow p_1 = \left\lceil \log_2 \frac{J_M}{T_{\text{packet}}} \right\rceil \\ &T_{\text{packet}} \leq 2^{p_2} T_{\text{cell}} \Rightarrow \; p_2 = \left\lceil \log_2 \frac{T_{\text{packet}}}{T_{\text{cell}}} \right\rceil \quad \Rightarrow \\ &p \; = p_1 + p_2 = \left\lceil \log_2 \frac{J_M}{c \times T_{\text{cell}}} \right\rceil + \lceil \log_2 c \rceil \; . \end{aligned} \tag{4.6.1.18}$$

It should be noted that the expression in Equation (4.6.1.18) is not the same as $\log_2 \frac{J_M}{c \times T_{\text{cell}}} + \log_2 c$ which simplifies to $\log_2 \frac{J_M}{T_{\text{cell}}}$, and therefore gives the same solution for p as the case of a single cell per packet displayed in Equation (4.6.1.17). If one attempts to use the same expression found in Equation (4.6.1.17) also for the case of multiple cells per voice packet, then the sequence number for the cells themselves

become dependent on the overall sequence number employed for synchronization. This will complicate the implementation because the functions of two different network layers will be intermixed. For this reason, we will only concern ourselves with the p_1 bits whose value is given by the same equation for both cases when $T = T_{\texttt{packet}}$ is considered.

4.6.2 Implementation

Synchronization between sender and receiver is achieved with the transmission of a p_1-bit sequence number. At the sender site, as shown in Figure 4.6.5, every time interval $T = T_{\texttt{packet}} = c \times T_{\texttt{cell}}$ a counter increments its value and executes modulo 2^{p_1}. The result is the p_1-bit sequence number for this time interval. If there is a useful voice data to be sent through the network, then these p_1 bits are sent over the network either along with the first cell of the packet or distributed in all of the cells of the packet. If there is no useful data to be sent, i.e. the sender is on a silence time interval, the counter is simply incremented and then executes modulo 2^{p_1}. The above procedure is repeated till the end of the conference.

Upon the reception of the packet, there are two processes that proceed in parallel by the receiver. The first process involves the post-processing of the arrival times of the packets. By exploiting the information of the arrival times of the packets alone, the receiver is able to determine the bounds of the network jitter, the expected average arrival time of the packets (the reference times), and the frequency offset between its clock and the clock at the sender. An additional advantage of this algorithm is that the sender is not required to transmit the generation time of the packet (time stamp) in order for the frequency offset to be determined by the receiver. The exact procedure of the first process related to the estimation of the frequency offset is investigated in the [52] and also presented in Chapter 6.

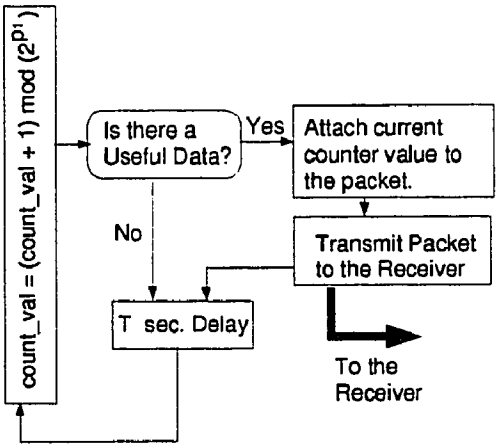

Figure 4.6.5 Sequence numbering and transmission of packets at the sender.

The second process is directly involved with the synchronization of the arrived packet by determining its correct playback time. In order for the second process to operate properly, the reference times, bounds of the network jitter, and the rate of the time offset must be known to the receiver by the first process. Because all of the above parameters need a certain number of pre-received packets, it is assumed that during the connection - setup time a few packets, let us say N, were sent prior to the execution of the synchronization algorithm. With their help, a rough estimation of the reference time of the last packet sent during the setup time $t_{ref,0}$ is made. These N packets sent by the sender carried among other useful information their own sequence numbers explicitly into their user field. This implies that the mapping $t_n \leftrightarrow n$ between the packet arrival times and the sequence numbers already exists for these N packets, and therefore the receiver using Equations (4.3.2), (4.3.3), and (4.4.5) can find the reference time of the last packet N, $t_{ref,N}$, which from now on we will call it $t_{ref,0}$ for convenience.

In Figure 4.6.6, a block diagram describing the actions of the two processes at the receiver is presented. The reader should notice the *local feedback* that couples the two processes. In order for the second process to determine the sequence number of each

arriving packet, the reference time of this packet must be known to the second process. However, in order for the first process to estimate the reference time of the arriving packet, the first process must have knowledge of the sequence number of the arriving packet. This is achieved with a feedback loop from the second process to the first as shown in Figure 4.6.6.

There are two counters at the receiver: one infinite counter whose value starts incrementing every T seconds right after the end of the setup time $t_{ref,0}$, and a finite counter whose value is repeated after a pre-determined number 2^{p_1}. In other words, $current = (previous + 1) \bmod 2^{p_1}$. The determination of the correct phase of the finite counter will be shown now. There is a p_1-bit sequence number associated with every packet which was attached to it by the sender during its departure. This *same* p_1-bit sequence number will be used to denote the arrival time of each packet at the receiver *as long as* the packet has experienced exactly the average delay. In other words, if the reference time of a packet n could be known to the receiver, then the reference time, and along with it the time interval $[t_{ref,n}, t_{ref,n} + T]$, will be assigned the p_1-bit sequence number this packet n carries with it. Then, since the network jitter the x packet experiences can be estimated using Equation (4.4.8), the receiver can determine the p_1-bit sequence number of the reference time of the x'^{th} packet $t_{ref,x} = t_x - \widehat{\Delta}_x$ as follows[2]

$$(t_{ref,x})_{p_1} := (x)_{p_1} \quad . \tag{4.6.2.19}$$

$(x)_{p_1}$ is the p_1-bit sequence number attached to packet x by the sender. Then the p_1-bit sequence number of the following k'^{th} interval away from $t_{ref,x}$, i.e the interval $[t_{ref,x} + kT, t_{ref,x} + (K+1)T]$, can be determined by the following operation

$$(t_{ref,x+k})_{p_1} = \left\{ (t_{ref,x})_{p_1} + k \right\} \bmod 2^{p_1} \; . \tag{4.6.2.20}$$

---

[2] := is the assignment operator.

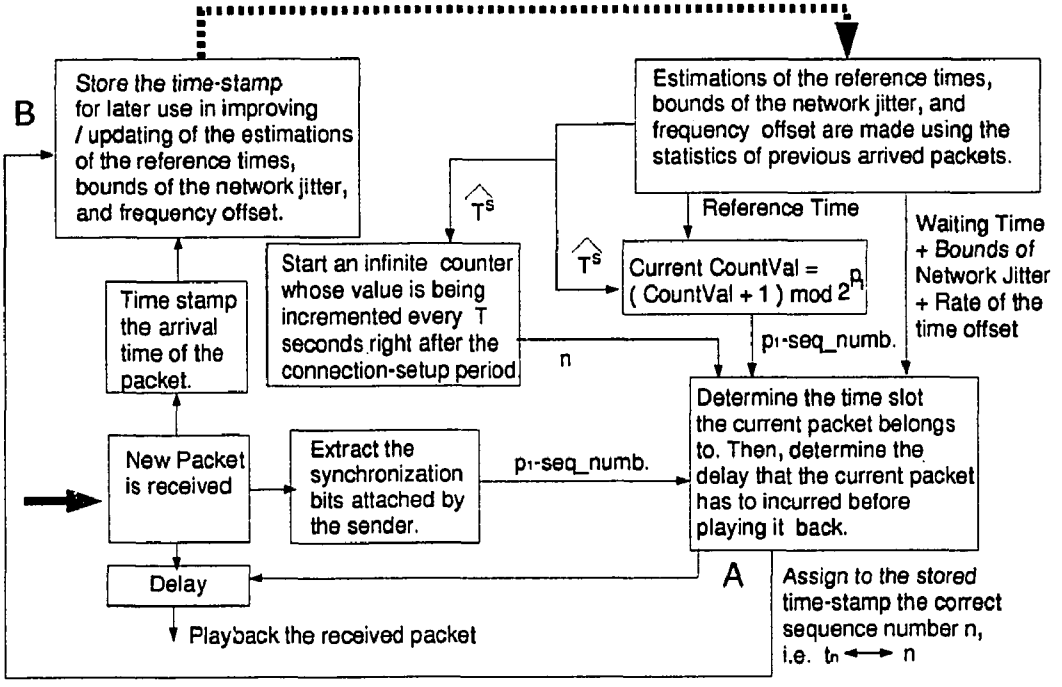

Figure 4.6.6 Block diagram of the receiver's actions upon the reception of a packet. Notice the feedback loop from point A to point B.

Once the reference times are mapped into p_1-bit sequence numbers, the p_1-bit sequence number of the current reference interval is compared with the p_1-bit sequence number of the packet that arrived within this time interval and needs to be synchronized. Synchronization now translates into a problem of finding the correct mapping between the arriving packet and its reference time. The reference time of the arriving packet is the beginning of the *nearest* interval T whose p_1-bit sequence number agrees with the p_1-bit sequence number of the arriving packet. The number δ of intervals T the reference time of the arrived packet is away from the beginning of the current interval is given by the following statement.

Theorem for determining the Reference Time of a Packet. *Theorem 2 Suppose an entire axis is divided into equally spaced intervals T. Each interval T is represented by a p_1-bit sequence number, such that the sequence number of the next interval is related to the previous by the following relation: next = (previous + 1) mod 2^{p_1}. A packet which carries an arbitrary p_1-bit sequence number falls into an arbitrary interval T. Given the p_1-bit sequence numbers of the packet $A = A_{p_1}A_{p_1-1}...A_2A_1$ and that of the interval T $B = B_{p_1}B_{p_1-1}...B_2B_1$ which fell into, then the nearest interval T whose p_1-bit sequence number $C = C_{p_1}C_{p_1-1}...C_2C_1$ exactly matches the p_1-bit sequence number of the packet is located δ intervals away (ahead —, behind +) from the current interval. δ satisfies the condition $A = (B + \delta) \bmod 2^{p_1} = C$ and is given by the following relation*

$$\delta = \sum_{i=1}^{p_1} \left(\bar{A}_i B_i - A_i \bar{B}_i\right) \times 2^{i-1} \qquad (4.6.2.1.21)$$

Proof: The proof of the above statement is based on the truth tables of the terms $A_i\bar{B}_i$ and $\bar{A}_iB_i$. $A_i\bar{B}_i = 1$ iff $A_i = 1$ and $B_i = 0$. Similarly, $\bar{A}_iB_i = 1$ iff $A_i = 0$ and $B_i = 1$. Therefore, if $A_i\bar{B}_i = 1$, then the contribution of this term to δ will be -2^{i-1}, etc. This ends the proof. □

It should be noted that the computations involved in determining δ using Equation (4.6.2.1.21) are extremely low, since only bit-wise operations are needed (bit-wise and, bit-wise complement, and (i-1) bit-wise shift for the multiplications).

Playback Subsequently, the proper time that the arrived packet should be played back is determined as follows

$$\texttt{playback}_{\texttt{time}}(n) = \widehat{t}_{ref,n} + W = (\texttt{current}_{\texttt{ref-time}} + \delta T) + W\,. \qquad (4.6.2.2.22)$$

$\texttt{current}_{\texttt{ref-time}}$ is the beginning of the interval the arrived packet fell into, and δ and W are given by Equations (4.6.2.1.21) and (4.5.14) respectively.

4.6.3 Example

In Figure 4.6.7, an example of a typical interaction between sender and receiver for synchronization is shown. The number of bits used in this example for sequence numbering is $p_1 = 3$. It is assumed that during the connection-setup time N packets were sent from the sender to the receiver and from those packets the receiver was able to make a rough estimation of the zeroth reference time $\widehat{t}_{ref,0}$, i.e. the reference time of the last N'$^{\text{th}}$ packet received. Then in both communicating entities (sender and receiver), a finite counter of length $2^3 = 8$ starts incrementing every T seconds. Packets at the sender will *always* be generated periodically and *in-phase* at the beginning of the interval T even though a silence interval is in between. The packets arrive at the receiver in times indicated as t_n, where n denotes the sequence number of the time the packet was generated at the sender. For example, the arrival time of the 4'$^{\text{th}}$ packet generated at the sender is denoted as t_6 and not t_4. By arriving at the receiver, each packet carries with it the 3-bit sequence number which was attached by the sender. The entire time axis at the receiver is also divided into $T = T^R$ time intervals that each one has its own 3-bit sequence number. The reference time for a packet is simply the beginning of the time interval T at the receiver that its 3-bit sequence number matches the 3-bit sequence number that the packet carries with it. Once the reference time of the arriving packet is found, its playback time will be W away from its reference time.

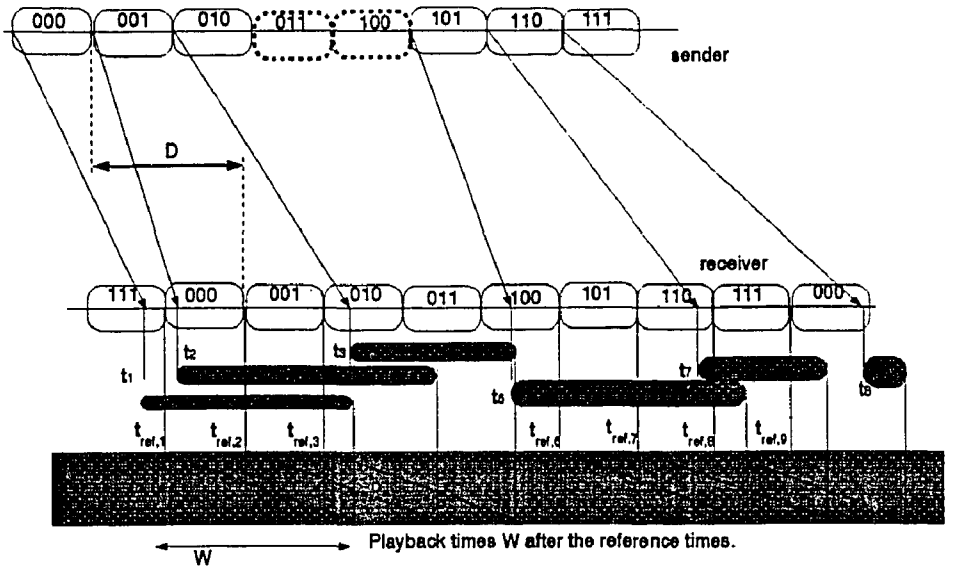

Figure 4.6.7 Example of a typical interaction between sender and receiver for synchronization.

In this example, let us have T = 6ms, and the maximum jitter from the average $\widehat{\Delta}_{max}$ observed in the network is 12ms. Allowing 2.5ms for the estimation error ϵ and the uncertainty in the time offset , the waiting time from the reference time becomes $W = 14.5ms$. If the current 3-bit sequence number of the reference time at the receiver shows 100 and during this time slot a packet arrived with 3-bit sequence number 101 (t_6), then using Equation (4.6.2.1.21), we have: $A_3A_2A_1 = 101$, $B_3B_2B_1 = 100$, $\Rightarrow$ $\delta = (\bar{1}1 - 1\bar{1})2^2 + (\bar{0}0 - 0\bar{0})2^1 + (\bar{1}0 - 1\bar{0})2^0 = -1$. δ= -1, means that the reference time of the arriving packet $\widehat{t}_{ref,6}$ is one time slot $T = 6ms$ into the future from the beginning of the current interval. Then, the playback time is simply $W = 14.5ms$ away into the future from the reference time of the packet $\widehat{t}_{ref,6}$.

4.6.4 Summary of the Synchronization Algorithm for Point-to-Point Single-Medium Communication

In this section, a summary of the synchronization algorithm described in this section is given. The main objective of this section is the determination of the correct playback time (or display time for a video packet) of a packet received, under the condition that

the synchronization is achieved with the minimum number of control bits. In this section, only the point-to-point single medium communication is addressed. However, the results presented are directly applicable to Intermedia Synchronization discussed in the next chapter, chapter 5, and in [45, 46], and to interparticipant synchronization investigated in [33, 32] and to be discussed in chapter 6. The steps of the synchronization algorithm are as follows:

1. During the connection setup time, the sender transmits to the receiver continuously and periodically N packets. All of these packets carry explicitly in their user field their own sequence number. Then using Equations (4.4.5) and (4.4.4), the receiver estimates the reference times (average arrival times) for each packet arriving.
2. Right after the connection setup time, both the sender and the receiver start a finite counter of length 2^{p_1}. If the sender has useful data to send, it attaches to the packet the p_1-bit sequence number of the current interval and then transmits the packet. At the receiver, the time axis is also divided into equally spaced intervals of time length T and each of them is assigned the current p_1-bit sequence number from the counter.
3. Using Equation (4.6.2.1.21), the receiver determines the average arrival time (reference time) of the arriving packet by matching the p_1-bit sequence number of the arriving packet with the *nearest* interval T with *identical* p_1-bit sequence number. Then the playback time is found by Equations (4.5.14) and (4.6.2.2.22).
4. The algorithm is executed as is until the error due to the estimation of the reference times and due to the uncertainty of the time offset in Equation (4.5.14) becomes larger than allowed. Then the algorithm is executed again, but this time the estimation of the reference times is achieved using Equations (4.4.8) and (4.4.11) which are properly suited for noncontinuous periodic traffic.
5. At all times, right after the determination of the reference time for each arriving

packet, the sequence number of each packet is also evaluated using an infinite counter which started the same time $\widehat{t}_{ref,0}$ as the finite counter. More about the functionality of this counter will be shown in a following paper dealing with the estimation of the frequency offset [52]. At the present, its usefulness will be on counting the received packets N and the time slots of duration T, $N+k$. Knowing this information, the receiver is able to evaluate the errors due to the estimation of the reference times (depending on N), and that due to the unknown time offset (depending on $N+k$).

4.7 Concluding Remarks for Chapter 3.

In this chapter, a synchronization algorithm for point-to-point real-time communication in a network environment was presented. For the case of point-to-point single medium communication, synchronization has the meaning of finding the proper playback time for each packet, proper in the sense that the playback time must be the time which is nearest to the minimum waiting time. The minimum waiting time for playback, as noted also from many other researchers, is the maximum network jitter observed in the network for this route.

The concept of the reference times (or average arrival times) which we introduced in [33, 32] (see also chapter 6) is also used in this chapter, but this time extended to encompass the case of noncontinuous periodic traffic. This is a significant improvement over our previous work because now a source which has no useful data to send at some intervals of time is not required to transmit "empty" packets in order to keep the receiver synchronized. A reader interested in the performance evaluation of the estimation process of the reference times is referred to our previous work [33, 32] or chapter 6. There, the design trade-offs among the various parameters of the estimation, *i.e.* the probability

$P(\epsilon)$ that the estimation of the reference time is outside a given confidence interval ϵ, the confidence interval ϵ itself, and the number of packets received N, are discussed in detail.

A large portion of this algorithm was devoted in determining the minimum number of bits in order for the receiver to uniquely identify the sequence number of the packet. This was achieved by exploiting the fact that a proper sequence numbering scheme should span a time duration not greater than the maximum jitter expected in the network. For the intermedia case, the time interval the numbering scheme needs to span is shown to be operationally the same as the one found for the point-to-point single medium communication (see [45, 46]).

The issue of the estimation of the frequency offset is discussed in [52] and in chapter 7. All of the above results are directly applicable to intermedia synchronization algorithm proposed in [45, 46] and presented also in chapter 5, and to the interparticipant synchronization algorithm proposed in [33, 32] and presented also in chapter 6.

Chapter 5 Intermedia Synchronization In Real-Time Multimedia Conferencing

5.1 Overview

Issues related to Intermedia Synchronization in a Multimedia Conference are discussed in this chapter. Specifically, two issues are explored in depth: optimal sequence numbering and minimum waiting time for playback. It is proven that the time interval the optimal sequence numbering scheme for Intermedia Synchronization should span needs not to cover a time duration greater than the maximum of the following two expressions: the maximum variation of the network jitter and twice the maximum absolute difference between the expected time delays of the two media. The concept of the reference times or expected arrival times that we have introduced in the previous chapter is also used in this chapter. As a consequence, many issues such as estimation of the reference times and bounds of the network jitter, are assumed to be known. Two different expressions for the minimum waiting time for the voice packets are derived: a minimum waiting time suitable when only the voice connection is active, and another when packets of both media are being transmitted. Finally, a switching mechanism that hunts for the minimum waiting time exhibiting the minimum delay suitable to the proper situation (voice transmission with or without video transmission) is presented.

One important application of interactive communication is multimedia conferencing. There are two fundamental classes of issues (problems) related to multimedia conferencing: management issues, and multimedia synchronization issues. The latter class deals with problems that arise when synchronization of packets from different participants or packets from different media (voice, video, text, etc.) is sought to be achieved. Multimedia synchronization problems are very closely related to the nature of computer

networks, which unlikely the telephone networks, are packet switching networks. In packet switching networks data is distributed in packets and each packet travels though the network independent of the other packets. While the packet switching concept is ideal for transferring data, i.e. text or e-mail, many problems arise when transfer of real-time data such as voice and video is required.

In this chapter, a subset of the Multimedia Synchronization issues that are specific to Intermedia Synchronization are addressed. Intermedia Synchronization deals with the problem of synchronizing packets emanating from different media sources without destroying the temporal relationships existed among them during playback at the receiver. Our approach to Intermedia Synchronization constitutes an integral part of our more general approach to Multimedia Synchronization. Specifically, in [33, 32] or Chapter 6 an algorithm was proposed for Interparticipant Synchronization, while in [53, 54] or Chapter 4 an algorithm was proposed for Point-to-Point Single Medium Synchronization.

During the last years, multimedia Synchronization has received a lot of attention. Synchronization confined to voice packets was investigated by Barberis and Pazzaglia in [15], Montgomery in [16], Felipe *et. al.* in [17], and [18]. Rangan and Ramanathan in [12, 13, 14] were the first to identify and give a solution to Interparticipant Synchronization without using any extra control bits for sequence numbering. Little, M. Woo, and Ghafoor in [36, 37, 55, 38, 39], L. Li *et. al.* in [40, 56], present various aspects of multimedia synchronization. C. Sreeman in [57] presents a service oriented approach to media synchronization, and C. Yang *et. al.* in [58] present another transport algorithm for multimedia synchronization using petri-nets.

In section 2, our algorithm for Intermedia Synchronization is presented. Two main issues are addressed. Sequence numbering using the optimal number of bits and minimum waiting time for playback for both media. In section 3 a switching mechanism which

looks for the minimum waiting time for playback for the voice packets is presented. In section 4 a summary of the Intermedia Synchronization algorithm is given, and in section 5 we present our concluding remarks.

5.2 Intermedia Synchronization Algorithm

In point-to-point network communication with a single medium, several problems were identified and an effort was devoted in solving them in the optimal way possible. The problems identified were the network jitter, sequence numbering scheme, and frequency offset. The problem dealing with the estimation of the frequency offset using both the least squares approach and our own approach is treated separately in [52] or Chapter 6. In our work in Chapter 6 about multiparticipant synchronization [33, 32] additional problems were identified such as the phase offset, different frequency offsets and bounds of the network jitter, and different average time delay for each of the connections. For intermedia synchronization it turns out that the same problems encountered for multiparticipant synchronization need to be addressed except the phase offset problem. Phase offset exists when two or more communicating entities transmit a signal to the same destination. Then, even though the periods for each of the transmitting signals are the same, there is going to be a difference in the starting points for each of the periods. For intermedia synchronization, since the data streams for different media are transmitted from physically the same host, it can be arranged so that the phase offset is zero and therefore it will not be considered in the analysis of intermedia synchronization. An additional complication which is not encountered in the multiparticipant case is the existence of different periods for different media. A valid assumption which we are making in this case is that the generation periods of different media are of integer multiples to each other.

In Figure 5.2.1 a general view of the routing for two different media streams (voice and video) is displayed. Due to different quality of service (Qo.S.) requirements, such as packet loss probability, bandwidth, and delay or processing requirements, the routes for the two media will most probably be different even though the departure and destination hosts are the same. This leads to the obvious statement that average time delays and bounds of the network jitter will be different for each media. Let us define the average time delay encountered by voice to be D^{vo} and the average time delay for the video streams to be D^{vi}. In addition, let the upper bounds of the network jitter for voice and video be $\left[-\Delta_{min}^{vo}, \Delta_{max}^{vo}\right]$ and $\left[-\Delta_{min}^{vi}, \Delta_{max}^{vi}\right]$ respectively. The superscripts in the reference times and the bounds of the network jitter denote whether the packet is video or voice. In this paper the same definition of the network jitter used in our previous works [33, 32, 53, 54] is followed. However, different definitions that are better suited in analyzing the phenomenon of network jitter with respect to the time delay and burstiness of the traffic are given by Cruz in [27, 48], Wang and Crowcroft in [29], Cidon *et. al.* in [49], and Matragi *et. al.* in [28, 50]. Finally, the generation periods of the packets for each media stream is T^{vo} for voice and T^{vi} for video, and in addition the two periods are assumed to be integer multiples with each other, i.e. $T^{vi} = z \times T^{vo}$, $z \in \mathsf{N}^{+}$.

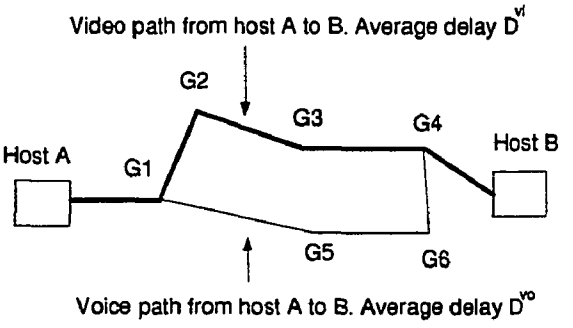

Figure 5.2.1 Routing for two different media streams (voice and video).

5.2.1 Optimal Sequence Numbering

The function of the Intermedia synchronization block is to insure that the temporal relations between different media (in our case only video and voice packets are considered) are preserved during playback at the receiver. It is intuitively clear that in order the temporal relations to be perfectly preserved the waiting time for both media streams should be adjusted so that the playback times coincide for packets that were generated at the same time at the sender. In Figure 5.2.2, a typical example of the probability density functions (*p.d.f.*) of the network jitter (as defined in [33, 32, 53] or Chapters 3 and 5) of the two media (voice and video) is presented. Let us assume that two packets, one from each media, were generated at the same time at the sender. Without loss of generality, we can also assume that they carry the same sequence number n. Then, as shown in [53, 54] and Chapter 3 where the Point-to-Point Communication Synchronization Block is discussed, the reference times $\widehat{t}^{vo}_{ref,n}$, $\widehat{t}^{vi}_{ref,n}$ for each packet and the maximum values of the network jitter $\widehat{\Delta}^{vo}_{max}$, $\widehat{\Delta}^{vi}_{max}$ for each of the media streams can be determined at the receiver. Then, for this example, the minimum waiting time for the video packets is simply $W_{vi} = \widehat{\Delta}^{vi}_{max} + \epsilon$, and for the voice packets is

$$W_{vo} = \widehat{\Delta}^{vi}_{max} + \epsilon + \left(\widehat{t}^{vi}_{ref,n} - \widehat{t}^{vo}_{ref,n}\right) = \widehat{\Delta}^{vo}_{max} + \epsilon + X \qquad (5.2.1.1)$$

ϵ designates an upper bound in the errors arising in estimating the reference times and the frequency offsets. The term X represents the extra time the voice packets have to wait for Intermedia Synchronization compared to waiting time voice packets would have had if synchronization with the video packets were not required. We would like also to make clear that the *minimum waiting time* for playback of the packets is defined as the minimum time packets from a particular connection have to wait after their corresponding

reference times till they are played back at the receiver. The minimum waiting time is a function of the packet loss probability desired. The smaller the packet loss probability, the larger the minimum waiting time, and in case a packet has not arrived within this time, the packet is considered lost. It should be noted that the difference of the expected time delays between the two media is given by

$$D^{vi} - D^{vo} = \hat{t}^{vi}_{ref,n} - \hat{t}^{vo}_{ref,n} \qquad (5.2.1.2)$$

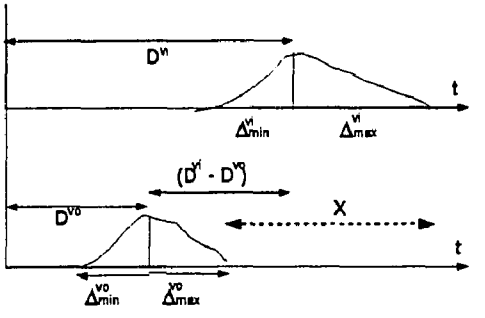

Figure 5.2.2 Voice packets have to wait an extra time interval X for playback for Intermedia Synchronization.

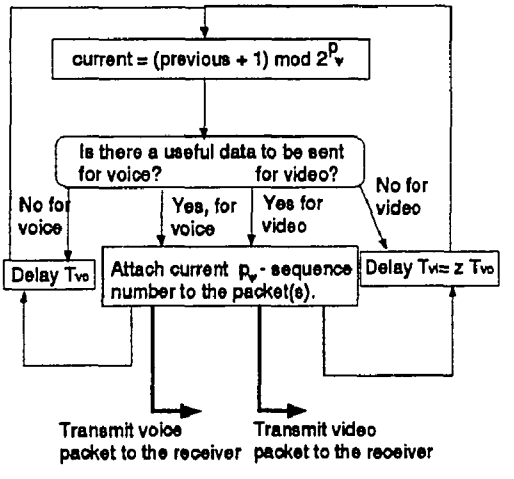

Figure 5.2.3 Sender's actions for Intermedia Synchronization.

It was made apparent from the example shown in Figure 5.2.2 and Equation (5.2.1.1) that in order the minimum waiting time for both media to be evaluated at the receiver, the difference of the expected time delays between the two media is needed. This implies that the receiver should be able to identify those pairs of packets —one packet from each medium— that they were generated at the same time at the sender. The easiest scheme is to have the sender attach the same sequence number to those pair of packets. An implementation of this scheme is shown in Figure 5.2.3. The time axis at the sender is divided into equal time intervals of time duration T^{vo}. Every time interval is mapped into a p_ψ-bit sequence number and the numbering of the intervals follows the rule $\texttt{current} = (\texttt{previous} + 1) \mod 2^{p_\psi}$. The sequence numbering of the time intervals is independent from the number of packets actually generated. In case a packet (can be a voice or a video packet) is generated, the sender attaches the p_ψ-bit sequence number of the current interval. There is one more bit (FLAG) that the sender attaches to the video packets. This FLAG bit is used by the sender to tell the receiver whether or not there is a voice packet generated at the same time with the video packet. More about the FLAG bit will be said later in this section. In order for the sequence number to be constructed with the minimum number of bits, the minimum time interval Ψ that the sequence numbering scheme should span should be found.

The ***requirements*** that the sequence numbering scheme for Intermedia Synchronization should fulfill are:

1. For each individual media stream, the receiver should be able to recognize the sequence number of a packet in any situation.
2. Assuming that packets, one from each media, that were generated at the same time at the sender carry the same p_ψ-bit sequence number, the sequence numbering scheme should correctly identify the packets that were generated at the same time at the

sender.

The time interval Ψ that fulfills the above requirements is given by the following theorem.

Theorem 1. (Reference times have been established for both media). *The time interval Ψ that the sequence numbering scheme should span for intermedia synchronization needs not to be greater than the following expression*

$$\Psi = \max\left\{2\left|D^{vi} - D^{vo}\right|_g\ ,\ (\Delta_{max})_g - (\Delta_{min})_g\right\} \qquad (5.2.1.3)$$

where $(\Delta_{max})_g = \max\left\{\left(\Delta_{max}^{vi}\right)_g\ ,\ \left(\Delta_{max}^{vo}\right)_g\right\}$ *and* $(\Delta_{min})_g = \min\left\{\left(\Delta_{min}^{vi}\right)_g\ ,\ \left(\Delta_{min}^{vo}\right)_g\right\}$. $\left|D^{vi} - D^{vo}\right|_g$ *represents the maximum difference in absolute value between the expected time delay of the video packets and the expected time delay of the voice packets one can observe in the network. It follows naturally that the number of bits needed to cover a time interval of duration* Ψ *is* $p_\psi = \lceil \log_2 \frac{\Psi}{T^{vo}} \rceil$.

Proff: The proof of Equation 5.2.1.3 is given in appendix B. □

In order for the receiver to identify a pair of packets that were generated at the same time at the sender, the pair exhibiting the minimum absolute difference of their reference times was chosen (see Equation (B.3)). However, in case no voice packet was generated at the same time with the video packet that we try to match in Equation (B.3), then applying Equation (B.3) will lead to a false result. For this reason, a 1-bit flag is added to the p_ψ-bit sequence number of the video packet to denote whether there is a voice packet generated at the same time with the video packet or not.

5.2.2 Minimum Waiting Time For Intermedia Synchronization

Using the methodology developed for point-to-point communication for a single medium and the numbering scheme elaborated in this section, the receiver is able to recognize, in the presence of noncontinuous periodic traffic, the following: 1) The reference times for each media, 2) The difference in expected time delays for the two

media , and 3) The bounds of the network jitter for each media. Then the general expression of the minimum waiting time suitable to Intermedia synchronization for the voice packets is

$$W_{vo} = \max\left\{\widehat{\Delta}_{max}^{vo} \ , \ (D^{vi} - D^{vo}) + \widehat{\Delta}_{max}^{vi}\right\} + \epsilon_R + \epsilon_f \ , \qquad (5.2.2.4)$$

and for the video packets is

$$W_{vi} = \max\left\{\widehat{\Delta}_{max}^{vi} \ , \ D^{vo} - D^{vi} + \widehat{\Delta}_{max}^{vo}\right\} + \epsilon_R + \epsilon_f \ , \qquad (5.2.2.5)$$

where ϵ_R is a bound in the error in estimating the reference times, and ϵ_f is a bound in the error arising from the frequency offset, i.e. $\max\{n_X \ , \ N + k - n_X\} \times \left|T^s - T^R\right| \leq \epsilon_f \equiv \max\{n_X \ , \ N + k - n_X\} \times 10^{-\nu} T^R$, where ν is a small number (from 3 to 6). For an explanation of the error bounds, the reader is referred to [53, 54].

Playback Time for Intermedia Synchronization In the synchronization block related to point-to-point communication (see [53, 54]), we have shown how the receiver can estimate the reference times of the arrived packets in the presence of noncontinuous periodic traffic. Further, a mechanism was presented that allows the receiver to make the proper mapping between the p_ψ-bit sequence number of the arrived packets and their corresponding reference times. Assuming that the reference times of the packets of both media are known to the receiver by appropriate actions of the Point-to-Point Synchronization block, the playback times for the voice and video packets are given as follows

$$\texttt{playback}_{\texttt{time}}{}^{\texttt{vo}}(n) = \widehat{t}_{ref,n}^{vo} + W_{vo} \qquad (5.2.2.1.6)$$

and

$$\texttt{playback}_{\texttt{time}}{}^{\texttt{vi}}(n) = \widehat{t}_{ref,n}^{vi} + W_{vi} \qquad (5.2.2.1.7)$$

5.3 Switching Mechanism for the Waiting Times of the Voice Packets

Since the length of a video packet is much larger than the length of a voice packet, processing and queuing delays will most probably make the expected time delay and the bound of the network jitter for the video packets larger than the respective delays for the voice packets ($D^{vi} > D^{vo}$, $\widehat{\Delta}^{vi}_{max} > \widehat{\Delta}^{vo}_{max}$). The reader is referred to our experimental work in [2] and in the works by Field *et. al.* [3] and Papadopoulos *et. al.* [59], where the expected time delays and the maximum bound of the network jitter are varying with the packet size. Therefore, the waiting time for the voice packets will be

$$W^1_{vo} = (D^{vi} - D^{vo}) + \widehat{\Delta}^{vi}_{max} + \epsilon_R + \epsilon_f \ , \tag{5.3.8}$$

rather than

$$W^2_{vo} = \widehat{\Delta}^{vo}_{max} + \epsilon_R + \epsilon_f \ . \tag{5.3.9}$$

where most probably $W^1_{vo} > W^2_{vo}$. Because the difference in the waiting times $W^1_{vo} - W^2_{vo}$ can be substantially large, quality of service will be improved proportionally if the first expression for the waiting time could be avoided whenever possible. In an application such as a multimedia conference held among two or more participants, not all of the connections will be active. This will be true for the voice connections because not more than one person at a time can speak. This is taken care automatically by the nature of the voice communication, because whenever a participant wants to speak and finds that someone else started already speaking, the conflict is resolved as in a typical face-to-face conversation.

The situation for the video connections is slightly different but this can be used to our advantage to improve the QoS of the voice transmission. For the transmission of the video packets, the user must generate a signal (probably by pressing a push-button in a

menu) indicating that he does not want to receive or send video data from / to a particular destination. This signal can be sent to all of the participants in the conference to make known that only voice data will now be transmitted. Therefore the minimum waiting time for the voice packets can be adjusted by the receiver as if the connection were single-medium (which is for the time being). We would like also to note that a situation may exist where a signal will be required to permit transmission of voice data. However, the important point is that a signal will be needed at all times for the transmission (or cease transmission) of the video data.

When a signal is received notifying the receiver that video data has ceased to be transmitted in a particular connection, the receiver will use the waiting time $W_{vo}^2 = \hat{\Delta}_{max}^{vo} + \epsilon_R + \epsilon_f$ from this point on for the voice packets. In order to avoid degradation of the voice quality, this adjustment should be made when the voice is during a silent interval. Here a theorem that permits the receiver to recognize whether a packet is the first packet of a talkspurt interval, and thus a silent interval has just ended, is presented.

5.3.1 Necessary and Sufficient Conditions for a Packet to Be the First Packet of a Talkspurt Interval.

Theorem 2. *Let us call the arrival time of the last packet received t_n, and the packet just previously arrived as t_m. Also, $t_{ref,n} > t_{ref,m}$. Then, the last packet received will be the first packet of a talkspurt interval if one of the following two conditions is true: 1) The difference of the arrival time of this packet with the arrival time of the packet just previously received is greater than the maximum variation of the jitter expected in this connection plus time interval equal to the generation period of the voice packets T^{vo}, and 2) In case the difference of the packet arrival times is less than the maximum variation of the network jitter plus T^{vo}, the last packet received can still be the first packet of a talkspurt interval if the p_ψ-bit sequence number it carries does not follow the rule*

$(\texttt{current})_{p_\psi} = \left[(\texttt{previous})_{p_\psi} + 1\right] \bmod 2^{p_\psi}$ *for consecutively generated packets at the sender. The two conditions are shown below*

$$\left(t_n - t_m > T^{vo} + \hat{\Delta}^{vo}_{max} - \hat{\Delta}^{vo}_{min} + \epsilon\right) \text{ or } \left((t_n)_{p_\psi} \neq \left[(t_m)_{p_\psi} + 1\right] \bmod 2^{p_\psi}\right) \quad (5.3.1.10)$$

Proof: The proof of the first part of the theorem is obvious since the biggest difference between two consecutively generated packets is $t_n - t_{n-1} = t_{ref,n} + \Delta^{vo}_n - t_{ref,n-1} - \Delta^{vo}_{n-1} = T^{vo} + \Delta^{vo}_n - \Delta^{vo}_{n-1} \overset{\max}{\rightarrow} T^{vo} + \Delta^{vo}_{max} - \Delta^{vo}_{min}$. The proof of the second condition is also obvious since for consecutively generated packets their p_ψ-bit sequence numbers should satisfy the condition $(\texttt{current})_{p_\psi} = \left[(\texttt{previous})_{p_\psi} + 1\right] \bmod 2^{p_\psi}$.□

5.3.2 Necessary and Sufficient Conditions for The Playback Times of Two Packets Not To Overlap

In order to avoid degradation in the voice quality, the adjustment of the waiting times should be done at this packet which is the first to be generated after a silent interval. However, one more condition should be met. The playback time of the first packet of a talkspurt interval which is intended to be played according to the new waiting time W^2_{vo} should be later or at least equal to the end of the playback time of the previously received packet, i.e.

$$(\texttt{playback}_{\texttt{time}}|_{W^2_{vo}})_n \geq (\texttt{playback}_{\texttt{time}}|_{W^1_{vo}})_m + T^{vo} \quad (5.3.2.11)$$

where $(\texttt{playback}_{\texttt{time}}|_{W^2_{vo}})_n$ indicates the beginning of the playback time of the $n^{'th}$ packet according to the waiting time derived by the expression W^2_{vo}. Since the playback time of a voice packet is W_{vo} (W^1_{vo} or W^2_{vo}) seconds into the future after the reference

time of the packet, the condition in Equation (5.3.2.11) can be rewritten as follows

$$\hat{t}^{vo}_{ref,n} + W^2_{vo} \geq \hat{t}^{vo}_{ref,m} + W^1_{vo} + T^{vo} \Rightarrow (n-m)T^{vo} \geq W^1_{vo} - W^2_{vo} + T^{vo}$$

$$\Rightarrow \quad n - m \geq \left\lceil \frac{W^1_{vo} - W^2_{vo}}{T^{vo}} \right\rceil + 1 \quad . \tag{5.3.2.12}$$

5.3.3 The Switching Mechanism

When the condition displayed on the second line of Equation (5.3.2.12) and the conditions of Equation (5.3.1.10) are satisfied, they will permit us to lower the waiting time for the voice packets without degradation of the voice quality. An implementation of the switching mechanism of the minimum waiting time for the voice packets is shown in Figure 5.3.4.

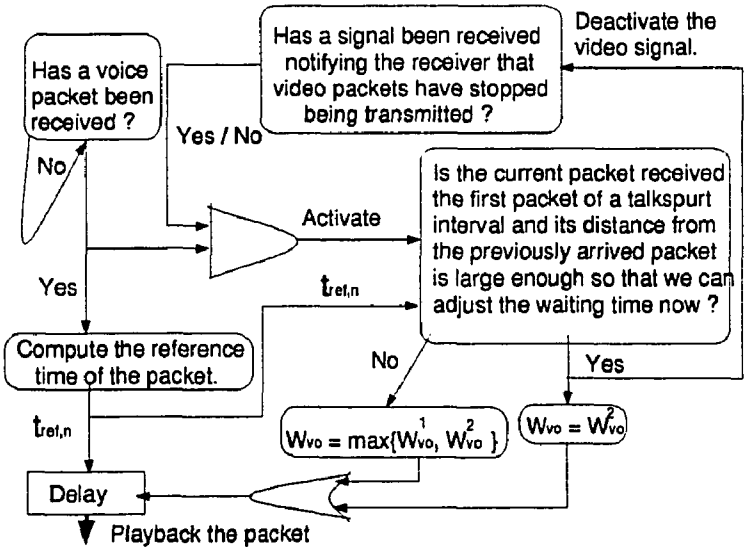

Figure 5.3.4 Switching scheme of the waiting time for the voice packets.

A similar switching scheme between the two waiting times can be implemented in case the receiver is been notified that a particular connection currently transmitting voice packets only, it will start transmitting video packets too. Since the switching of

the waiting times in this direction can only lead to an increase in the time delay, only the conditions in Equation (5.3.1.10), i.e. that the packet received is the first packet of a talkspurt interval, are needed. In Figure 5.3.5, a block diagram of the switching mechanism is presented. The receiver is in one of two states: in state one, only voice packets are transmitted, and in state two both voice and video packets are transmitted. Thus, when the receiver is in state one and a signal is received that video packets will be transmitted too, the receiver goes to state two and the minimum waiting time for the voice packets is the minimum waiting time derived for Intermedia Synchronization (see Equation (5.2.2.4)). If , while the receiver is in state two, the receiver is notified that video packets will stop being transmitted, the receiver goes back to state one and the minimum waiting time for the voice packets is identical to Point-to-Point Single-Medium Communication (see [53, 54]). The minimum waiting time for the video packets is simply given by Equation (5.2.2.5) and no switching mechanism is implemented for the following two reasons: 1) Most of the time, the maximum time delay will occur for the video packets and not for the voice packets, and 2) No echo problem exists for video data.

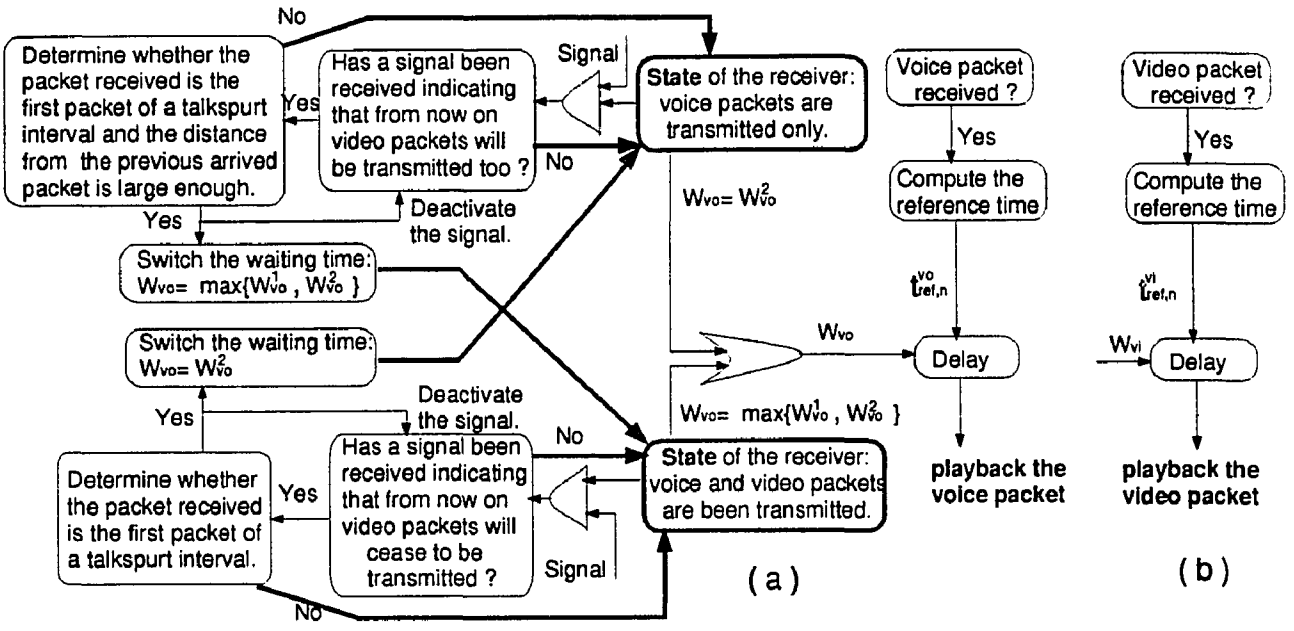

Figure 5.3.5 Receiver's actions in determining the playback times for a) the voice packets, and b) the video packets.

5.4 Summary of the Intermedia Synchronization Algorithm

The Intermedia Synchronization Algorithm proposed in this Chapter constitutes the second block of a three-block algorithm-protocol proposed for Multimedia Synchronization. The other two blocks being the Point-to-Point Single Medium Communication and Interparticipant Synchronization. In this Chapter we assume that the reader has knowledge of our previous work , especially the one appearing in [53, 54] and in Chapter 3 about the synchronization block related to Point-to-Point Single Medium Communication. The steps of the Intermedia Synchronization block at the receiver are as follows:

1. The reference times and the bounds of the network jitter are determined by the first synchronization block, the block related to Point-to-Point Single Medium Communication. In addition, since the time interval Ψ the sequence numbering scheme spans is $\Psi = \max\left\{2\left|D^{vi} - D^{vo}\right|_g , (\Delta_{max})_g - (\Delta_{min})_g\right\}$, the Point-to-Point Synchroniza-

tion block is proven that can identify the sequence numbers of the arriving packets for both media (voice and video).

2. The time interval Ψ of the sequence numbering scheme is also proven that it is sufficient to distinguish packets from different media that were generated at the same times at the sender. In other words, it is shown that in case $\left|t^{vi}_{ref,n} - t^{vo}_{ref,n}\right| = \min_{m \in \mathfrak{M}} \left\{\left|t^{vi}_{ref,n} - t^{vo}_{ref,m}\right|\right\}$, then this pair of packets with reference times $\widehat{t}^{vi}_{ref,n}$ and $\widehat{t}^{vo}_{ref,n}$ were generated at the same time at the sender.

3. Once packets that were generated at the same time can be identified, the difference in the expected time delays between the two media is given by $D^{vi} - D^{vo} = \widehat{t}^{vi}_{ref,n} - \widehat{t}^{vo}_{ref,n}$.

4. The minimum waiting time for the video packets is then given by $W_{vi} = \max\left\{\widehat{\Delta}^{vi}_{max}\ ,\ D^{vo} - D^{vi} + \widehat{\Delta}^{vo}_{max}\right\} + \epsilon_R + \epsilon_f$, and for the voice packets is $W_{vo} = \max\left\{\widehat{\Delta}^{vo}_{max}\ ,\ (D^{vi} - D^{vo}) + \widehat{\Delta}^{vi}_{max}\right\} + \epsilon_R + \epsilon_f$.

5. If $\widehat{\Delta}^{vo}_{max} < D^{vi} - D^{vo} + \widehat{\Delta}^{vi}_{max}$, and video packets have stopped being transmitted, then the minimum waiting time for playback for the voice packets should be $W_{vo} = \widehat{\Delta}^{vo}_{max} + \epsilon_R + \epsilon_f$.

6. The playback times for the voice packets is $\texttt{playback}_{\texttt{time}}{}^{\texttt{vo}}(n) = \widehat{t}^{vo}_{ref,n} + W_{vo}$ and for the video packets is $\texttt{playback}_{\texttt{time}}{}^{\texttt{vi}}(n) = \widehat{t}^{vi}_{ref,n} + W_{vi}$.

5.5 Concluding Remarks for Chapter 4.

An algorithm for Intermedia Synchronization was proposed in this Chapter. Not all issues relevant to Intermedia Synchronization are discussed here because many of them have been addressed in Chapter 4 and in [53, 54]. The proposed Intermedia Synchronization Algorithm constitutes the second block of our Multimedia Synchronization Algorithm. The first block is the synchronization block related to Point-to-Point Single-

Medium Synchronization and the third block is related to Interparticipant Synchronization (see Chapter 6 or [33, 32]). A big advantage of our algorithm is its modularity because the operation of each block is independent of the operation of the other two blocks. Thus, the Intermedia Synchronization block uses the results of the first block but it acts completely independent. As we have shown in this paper, the sequence numbering scheme needs not to be changed in order to encompass the case of Intermedia Synchronization, if the time interval Ψ the sequence numbering scheme spans is given by $\Psi = \max\left\{2\left|D^{vi} - D^{vo}\right|_g , (\Delta_{max})_g - (\Delta_{min})_g\right\}$. It is very important to notice that the expression for Ψ for Point-to-Point Single-Medium Synchronization, operationally is not different than that of the Intermedia since $2\left|D^{vi} - D^{vo}\right|_g \approx (\Delta_{max})_g - (\Delta_{min})_g$. Similar comments are valid for the optimal minimum waiting time that packets have to wait after their corresponding reference times in order to be played back. In this paper we have introduced a switching mechanism that switches between the two waiting times for the voice packets without degrading the quality of service. Thus, according to whether only voice packets or packets from both media (voice and video) are transmitted, the receiver is switching between the two minimum waiting time expressions derived for the voice packets. One expression suitable to the case that only voice packets are transmitted through a particular connection, and the other for the case that voice packets are required to be synchronized with their corresponding video packets (Intermedia Synchronization). Since the multimedia synchronization algorithm is based on the arrival times of the packets and not on their generation times, the addition-deletion of a new participant in the multimedia conference will have no effect to other participants.

Chapter 6 Interparticipant Synchronization in Real-Time Multimedia Conference Using Feedback

6.1 Overview

In this chapter, synchronization issues arising in a multimedia multi-party conference are addressed. Specifically we propose an algorithm to determine the set of packets generated periodically from different participants that are arriving at a node, either for mixing at the master of a conference, or for simply playing back at a regular participant of a conference. No global synchronization of the clocks is assumed. In this work the statistical approach rather than the deterministic is used to determine the proper set of packets that have to be mixed at a given time. The statistics are derived from the time stamp which each packet carries along with it. The essence of the proposed algorithm is to estimate the average packet arrival time (or reference time) for each participant. With the reference time at hand, the maximum jitter and the optimum waiting time for a mixer to wait packets from all participants can be determined. Both the accuracy and the computational complexity of the algorithm are improved by predicting the time/frequency offsets existing between the clocks at the source and the mixer. By employing feedback, the proposed algorithm is shown to lead to an optimum waiting/delay time. It is optimum in the sense that it is the same waiting/delay time that packets would experience in the case of only two participants, i.e. one sender and one receiver. The error of the proposed algorithm is enumerated by the Chernoff bound and demonstrated by simulation, and is shown to be acceptable in practical application. The algorithm can also be employed when traffic sources operate with different periods.

The second class of synchronization problem which we called *interparticipant synchronization*, is related to synchronization of packets arriving from different sources and is the subject of this chapter. As shown in Figure 6.1.1, receiver R can be any of the participants or a separate mixer and as noted both the average time delay D^i and network jitter are different from participant to participant. In this Chapter, the existence of different bounds in the arrival time of a packet from different sources is recognized to improve the synchronization algorithm. When a reference to packets is made we really mean the messages from the application level. In the paper authored by Rangan *et al* [12] (their synchronization algorithm has been proved to be optimum in the case there is no exchange of any control messages) the deterministic approach is followed whereas the statistical approach is taken in our work. A similar problem, the problem of voice synchronization between sender and receiver has been studied by many researchers [15, 16, 17].

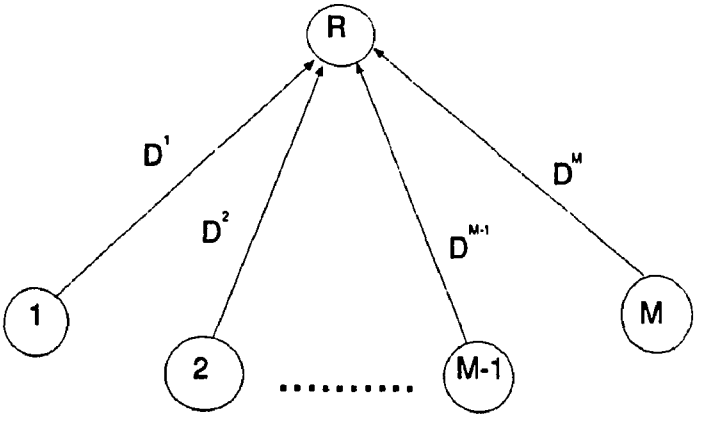

Figure 6.1.1 Multi party conference with M+1 participants.

The packets from a multimedia participant are assumed periodically generated. This assumption is an unrealistic one especially for voice traffic where the phenomenon of silence-talkspurt intervals exists. The problem of multimedia synchronization in the presence of silence-talkspurt intervals is dealt in the previous chapters. Along with

each packet its generation time at the source is included. However, if the period of the sources is known to the receiver it suffices to send the sequence number of the packet. Throughout this paper a reference to a time stamp will mean both the actual time stamp or the sequence number of the packet. Knowing the sequence number of each packet is fundamental to interparticipant synchronization for the following reasons: firstly, one can determine easily when a packet has arrived out of order; secondly, once the proper sequence of the packets is determined, the algorithm needs to be executed only periodically. To illustrate the latter point, suppose a multimedia conference is held among five (5) participants. Interparticipant synchronization in this chapter and in the works of Rangan and Ramanathan in [12, 13, 14] has the meaning of determining the set of packets from the different participants that will be mixed (at a master of a video conference) or played back/displayed (at a regular participant) at the same time. It is clear that only one packet from each participant belongs in the same set. Let us also assume that at some point in time interparticipant synchronization was achieved at one of the participants. This implies that the set of the packets (arriving from the other four participants) has been determined. Further, let us assume that the sequence numbers of the packets belonging to this set are as follows: 45 for the first participant, 42 for the second, 45 for the third, and 46 for the fourth. Then, at the next playback interval T, the new set will be comprised with packets having sequence numbers 46, 43, 46, and 47. In case there is no frequency offset between the clocks at the sources and the clock at the receiver, then interparticipant synchronization would have been achieved for the rest of the time. However, due to existence of frequency offsets between the clocks, synchronization is shown to be achieved only for a certain interval of time whose duration is to be determined.

The rest of the chapter is as follows. In section 2, the definitions of the network

jitter and of the reference times are presented. In section 3 , we present a new interparticipant synchronization algorithm. We identify the problems in relation to multimedia conferences and present the part of our algorithm which estimates the average arrival time of a packet for each source. Then, the interval $[\Delta^i_{min}, \Delta^i_{max}]$ for each source i and the minimum waiting time for the reception of one packet from each of the sources is determined. A modification of the algorithm is also presented for the case of sources transmitting packets with different periods, the periods being integer multiples of the fundamental period T_f .

In section 4 , we present a methodology to predict the timeshifts of the average arrival time of a packet for each source relative to the clock at the receiver, and find bounds of the probability of error in our predictions. Next, a more elaborate performance analysis using the mean square error as a criterion is presented which shows the optimality of our algorithm. In section 5 we describe how feedback can be employed to lead to an optimum waiting/delay time. In section 6 performance bounds of the estimation error in calculating the average arrival time and numerical examples of various parts of the interparticipant synchronization algorithm are presented. Finally, in section 7, we present our concluding remarks.

6.2 Definition of the Network Jitter and the Reference Times

The variability of packet arrival time is also known as the *network jitter* and we define it as the variation between actual arrival time and average arrival time. As shown in Figure 6.2.2, D^i is the average time delay of packets from a source i to arrive at the destination and T is the generation period of the packets at the sender. The actual arrival time of the $n'th$ packet from source i is denoted as t^i_n, where n is an integer. If there is no variation in the delay, the packets would arrive at points $t^i_{ref,n}$ indicated

in Figure 6.2.2, and observe that

$$t^i_{ref,n} = t^i_{ref,1} + (n-1)T^i \ , \ n = 1,2,...$$

$$or \tag{6.2.1}$$

$$t^i_{ref,n_1} = t^i_{ref,n_2} + (n_1 - n_2)T^i \ , \ n_1, n_2 = 1,2,...$$

where T^i is the transmission period T of source i measured according to the clock at the receiver. The *nominal* value of the transmission period of the packets measured at the clock at the source i is T. A good reference for internet time synchronization is the paper by Mills [21]. Unless explicitly written, when we compute the set of points defined by Equation (6.2.1) we will simply use T=T^R=T^i because the observer sitting in the receiver only knows time according to his own clock. The distinction we are making in Equation (6.2.1) using T^i instead of T will prove versatile when time/frequency offsets are considered later in this paper. Since packets experience a variation in the delay, an alternative definition for the reference points $t^i_{ref,n}$ will be as follows: the reference times for source i are defined to be the set of points generated by Equation (6.2.1) with the property

$$\lim_{N-\infty} \frac{1}{N} \sum_{n=1}^{N} \left(t^i_n - t^i_{ref,n}\right) \rightarrow 0. \tag{6.2.2}$$

In other words, the sum of the differences between the actual packet arrival times and the corresponding reference points approach zero as N approaches infinity. Consequently, the network jitter Δ^i_n for the n'th packet received from source i is defined as the difference between these two points, i.e.

$$\Delta^i_n \ = t^i_n - t^i_{ref,n} = t^i_n - \left(t^i_{ref,1} + (n-1)T^i\right) \ . \tag{6.2.3}$$

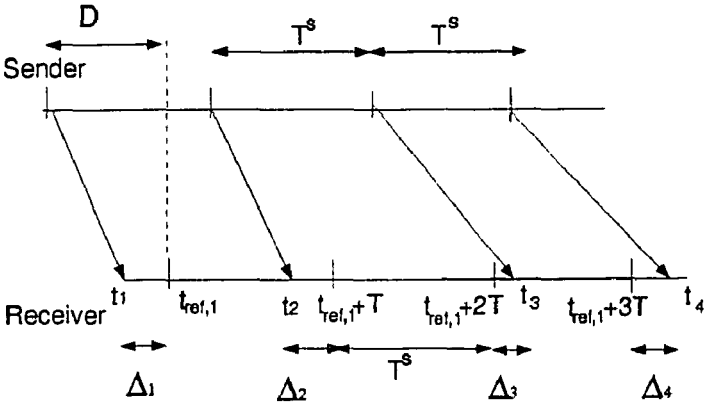

Figure 6.2.2 Determination of network jitter Δ^i_n for each packet n sent from source i.

6.3 Interparticipant Synchronization

6.3.1 Issues

A participant in a multimedia conference may receive packets from two or more participants. Each packet contains time sensitive data such as voice packets and an algorithm is needed to counteract any time asynchrony. For example, voice can be distorted for the following reason. In order to listen to all of the participants at the same time, the voice packets have to be mixed together into one and is then to be played back. If playback is not sufficiently smooth caused by the presence of network jitter, a degradation of the voice quality is observed. Degradation of voice quality occurs when either one packet from a specific source arrives late and is not played along with the rest, or a packet arrives out of sequence (If unreliable transport protocol, for example *UDP*, is used).

Below we identify most of the issues related to interparticipant synchronization that will help us put the problem into the right perspective:

1. The time delay for each packet might be different from one source to another, but as long as the difference is preserved, quality of service will not be affected.

2. Even though different participants begin to send their packets with the same period T, phase offsets of their starting times are inevitable due to the lack of perfect synchronization among participants.

3. Packets belonging to the same set will not be received at once but most likely in a time window whose length will be influenced by the phase offsets of the sources, the frequency of the clocks and the network jitter.

4. If interparticipant synchronization at a time instance is achieved and the exact frequency of the clocks of all the sources is known to the receiver, synchronization can be kept for the rest of the time.

6.3.2 Summary of the Interparticipant Synchronization Algorithm

In this chapter, the maximum error allowable in the synchronization of packets from different sources will be designated ϵ_f. As it will be shown later, the error will be composed from two components : one error component due to the finite number of packets used to estimate the reference times ϵ_R, and another due to the time offset acquired in the time interval NT (ϵ). Summary of the synchronization algorithm is provided now.

Main Algorithm The steps below are executed every new N packets received from each source i. Synchronization is shown to be achieved for the next N received packets.

a. Determine the reference time and maximum and minimum values of the network jitter ($\Delta^i_{max}, \Delta^i_{min}$) for each source i .
b. Divide time at the receiver in equally spaced intervals of duration T. These intervals are named the reference intervals, and packets whose reference times fall into the same reference interval belong to the same set.
c. Determine the additional time (minimum waiting time) after the end of the current reference interval that the receiver has to wait in order to receive one packet from

each source, i.e. to receive all packets belonging to the current set. The waiting time for each set is the time measured from the end of its reference interval to its playback time.

d. Packets belonging to the same set are played back after a waiting time equal to the one found in step c.

Enhancement to the Algorithm If steps a, b, c and d are repeated 2k times, 2kN packets are received from each source. This additional information can be utilized to improve the synchronization algorithm. The steps are:

a) Estimate the frequency offsets between the clock at each source i and the clock at the receiver.

b) Determine the prediction interval, which is defined as the interval after the time 2kNT that the error in the estimation of the frequency offsets will be less than the error ϵ.

c) Estimate the time that it will take for the reference time of each source i to move out of its reference interval (time ηT). This time has to be shorter than the prediction interval in order for the error of the estimation to be less than ϵ. This is the time instance that packets from source i when arrive at the receiver will be played back with a different set of packets, either one ahead or one behind.

Feedback Feedback can be employed both before or after the start of the prediction algorithm. Employing feedback reduces the waiting/delay time that each packet experiences to the minimum. The basic steps of the feedback algorithm are:

a) Determine the distance in time that the reference time of each source i is away from a point M (we chose it to be the middle of T, but any other point will do) and send this information back to each source i.

b) Upon reception of the feedback packet, the source i either pauses time d^i or inserts noisy data of time duration d^i in the first packet depending on whether its clock is faster or slower than the clock at the receiver.

During the conference setup time, we may allow the error to be larger than ϵ. As will be shown later in section 5, by allowing the larger error, the number of packets N needed for a certain level of accuracy is decreased geometrically. An alternative scheme will be to transmit with a shorter period $T_n = \frac{T}{n}$ in order to achieve higher accuracy.

6.3.3 Reference Time Estimation

Since the reference time plays vital role in our synchronization algorithm, we first focus on finding the best estimate of the reference time. The reference time is defined as average arrival time of the packets at the receiver from a multimedia source. Reference times are important because these will be used in determining whether packets from different sources belong to the same set for mixing or not. They have the characteristic to be relatively immune to network jitter provided the averaging process is done over sufficient number of packets. To estimate the reference time for a particular source i, $\widehat{t}^{\,i}_{ref,n}$ (hereafter, estimated values will appear with a $\widehat{\ }$), let the arrival time of the $\lceil \frac{N}{2} \rceil \; th$ packet (Because Ncan be either even or odd, we use the notation of the ceiling, i.e. the smallest integer greater or equal to the argument.)$t^i_{\lceil \frac{N}{2} \rceil}$ be the reference time for this source. We will show later in this section that the choice of the $\lceil \frac{N}{2} \rceil \; th$ packet arrival time $t^i_{\lceil \frac{N}{2} \rceil}$ in Equation (6.3.3.5) results in cancellation of the time offsets, and therefore, the proof is valid even for this case. Until the $\lceil \frac{N}{2} \rceil \; th$ packet arrives, we simply register the arrival times t^i_n , $1 \leq n \leq \lceil \frac{N}{2} \rceil - 1$, of the previously received packets. When all N packets have been received, we measure the time differences between the arrival times of the rest of the packets from this source and the set of points defined by

$$t^i_{\lceil \frac{N}{2} \rceil} + \left(n - \left\lceil \frac{N}{2} \right\rceil \right) T^R \; , \; 1 \leq n \leq N \; . \tag{6.3.3.4}$$

After all the differences are added up, and the entire sum is divided by the total number of the received packets N, we estimate the time $\Delta^i_{\lceil \frac{N}{2} \rceil}$ our initial guess $t^i_{\lceil \frac{N}{2} \rceil}$ is off from the actual reference time, which is given by

$$\widehat{\Delta}^i_{\lceil \frac{N}{2} \rceil} = \frac{1}{N} \sum_{n=1}^{N} \left(t^i_{\lceil \frac{N}{2} \rceil} + \left(n - \left\lceil \frac{N}{2} \right\rceil \right) T^R - t^i_n \right) \; . \tag{6.3.3.5}$$

Detailed proof of Equation (6.3.3.5) and the reason why the packet arrival time $t^i_{\lceil \frac{N}{2} \rceil}$ is used and not any other packet arrival time is given in Appendix C.

With $\widehat{\Delta}^i_{\lceil \frac{N}{2} \rceil}$ available, the estimated reference time for source i is obtained as

$$\widehat{t}^i_{ref,n} = \widehat{t}^i_{ref,\lceil \frac{N}{2} \rceil} + \left(n - \left\lceil \frac{N}{2} \right\rceil\right) T^i = t^i_{\lceil \frac{N}{2} \rceil} - \widehat{\Delta}^i_{\lceil \frac{N}{2} \rceil} + \left(n - \left\lceil \frac{N}{2} \right\rceil\right) \left\{ T^R - \left(T^R - T^i\right) \right\} . \tag{6.3.3.6}$$

The maximum and minimum bounds of the network jitter can also be obtained as

$$\widehat{\Delta}^i_{max} = \max_n \left\{ t^i_n - \widehat{t}^i_{ref,n} \right\} , \; 1 \leq n \leq N \tag{6.3.3.7}$$

and

$$\widehat{\Delta}^i_{min} = \min_n \left\{ t^i_n - \widehat{t}^i_{ref,n} \right\} , \; 1 \leq n \leq N \tag{6.3.3.8}$$

We cannot assume that the maximum jitter is known a priori for a given protocol because network jitter is affected by random network traffic and routes taken. Obviously there exists a trade-off between the accuracy of the algorithm and the number of the packets taken into consideration. This trade-off will be elaborated later in section 6.

6.3.4 Minimum Waiting Time For Playback

The minimum time window required for the reception of one packet from each source (i.e. 1 packet from source 1, 1 packet from source 2, ... , and 1 packet from source M) generating packets with period T in the absence of network jitter is T. This is due to the phase differences among the sources. In [12], it is stated that the smallest time window during which all M sources are guaranteed to generate packets is $T - \frac{T}{2^{M-1}}$. However, in practice due to the frequency drifts of the clocks, we will see that this minimum

time window requires constant movement at the receiver which is not desired because of complexity in implementing the algorithm. Here is the usefulness of the reference time which is relatively immune to the network jitter and therefore indicates with high probability of confidence whether a packet received belongs to the set whose reference time is within a time window or not. Packets with reference times belonging to the same time window T will be assigned to the same group (in [12] it is called the *Fusion set*) when played back at the receiver. In contrast to the reference time, actual packet arrival time varies due to network jitter. A packet from source i with its reference time at the end of the time window T and having maximum jitter Δ^i_{max} , might arrive as late as $T + \Delta^i_{max}$. The minimum waiting time to receive one packet from each source can be shown to be

$$W_{min} = T + \max_i \left\{ \widehat{\Delta}^i_{max} \right\} + \epsilon + \left| T_{off} \right| , \ 1 \leq i \leq M \qquad (6.3.4.9)$$

The fourth term $\left|T^i_{off}\right|$ is due to the maximum time offset (In any real network there are mechanisms that re-adjust the time offsets of each clock to the timehost of the network e.g. the unix rdate command. This possibility should be considered for any actual implementation of our algorithm, but for the sake of clarity we do not consider the implications of this problem in this work.) we expect from the different clocks at a particular situation. The total error, given as a design specification as noted before, will be distributed in two components: the time offset $|T_{off}|$ and the estimation error ϵ. The proportionality between these two components will be determined by the designer of the algorithm to properly suit his environment. The time offset is defined as

$$T^i_{off}|_{t=nT} = \left(f^i - f^R\right) nT \qquad (6.3.4.10)$$

where f^i and f^R are the frequency of the clocks at source i and at the receiver. The error due to unknown (yet) time offset will be removed from Equation (6.3.4.9) when the prediction algorithm described in section 4 is applied.

As it is shown in Figure 6.3.3, the minimum waiting time consists of two intervals. Interval AB (=T) and interval BC. Interval AB is due to the phase offset and is the interval which often will be referred to as the reference interval, and interval BC is function of both the network variation in the delay and the frequency differences of the clocks between sender and receiver. Note that interval BC overlaps with the next interval BD. A packet whose reference time is in interval AB and arrives at interval BC will be played back at time C. However, a packet whose reference time is in the interval BD and arrives at interval BC will be played back at time E.

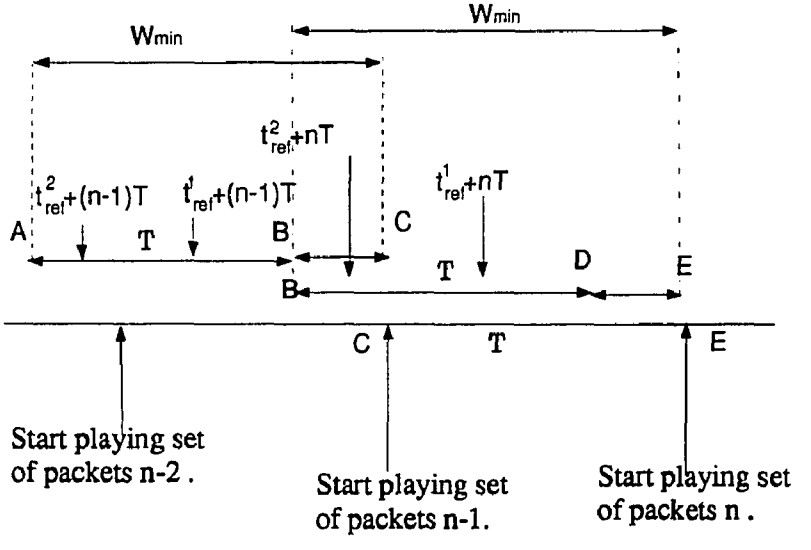

Figure 6.3.3 Minimum Waiting Time.

6.3.5 Minimum Waiting Time for Sources With Different Periods

Here we modify the minimum waiting time that will permit the use of sources transmitting packets with different periods. The periods are assumed to be integer multiples of the shortest packet generation period T_f. The subset of sources transmitting packets with period T_f will be referred to as the fundamental subset f. The minimum waiting time for *all* sources will be

$$W_{min} = T_f + \max_i \left\{ \hat{\Delta}^i_{max} \right\} + \epsilon + \left| T_{off} \right| , \; 1 \leq i \leq M \qquad (6.3.5.11)$$

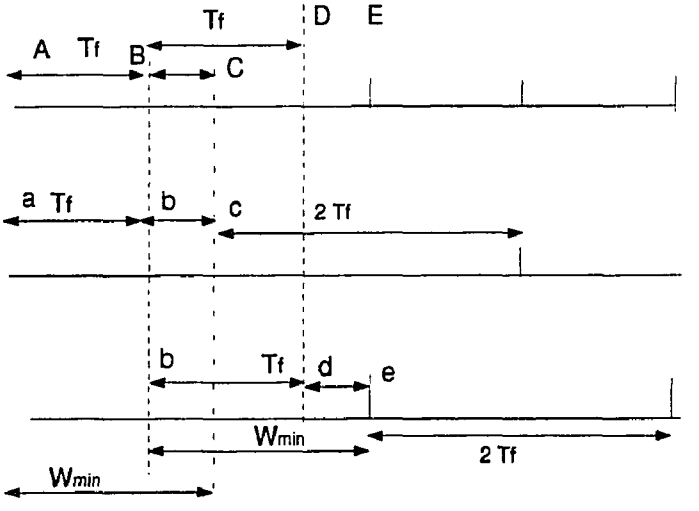

Figure 6.3.4 Minimum waiting time for sources with multiple packet generation periods.

The validity of Equation (6.3.5.11) is deduced from the following syllogism. It is clear that in the absence of other sources with multiple packet generation periods, Equation (6.3.5.11) holds at least for subset f. The reference time of a source with larger packet generation period, let's say nT_f, has to fall within one interval T_f of subset f. The next packet is not expected to arrive until a time interval nT_f has elapsed.

Consider for each reference interval T_f all the sources that their reference times fall into. With the exception of the range i, which will have to be a member only from the sources with reference times within this interval, the minimum waiting time will be as defined in Equation (6.3.5.11). By repeating the above until all the sources have been exhausted, a different source i can be found for different reference intervals. Different i will cause a problem when the reference time of a source falls into another reference interval and this is the reason for considering i to be any source in Equation (6.3.5.11).

The first line in Figure 6.3.4 shows sources with period T_f, while in the two lines below that, there are sources with periods $2\ T_f$. So sources with period $2\ T_f$ and reference times falling within time interval [a,b] are played back at time c (= C). If their reference times fall within time interval [b,d], they are played back at time e(= E).

6.4 Prediction of Resynchronization Interval

This section constitutes an enhancement to the *Interparticipant Synchronization Algorithm* described in section 3. The algorithm described so far can work by itself synchronizing packets from different sources every NT seconds. The only drawback of the algorithm described in section 3 is that we do not have exact knowledge of the frequency drifts of the clocks yet and therefore the worst case scenario has to be assumed in the determination of the time offsets in Equation (6.3.4.9).

As time progresses, more and more packets arrive, improving the accuracy of the algorithm and making it possible to use the methodology described in this section for estimating the frequency differences between the clocks at the sources and the clock at the receiver. Knowing the frequencies, how the reference times move as function of time can be determined. This in effect means knowledge of the time instance that a particular reference time will move out of its reference interval and therefore fall into another group

or Fusion Set. Since this is the function of the interparticipant synchronization algorithm, the algorithm itself becomes redundant once the frequencies of the clocks are known. The reader is referred to section 3.1, statement 4, where it is noted that once synchronization is obtained at some time instance and accurate knowledge of the frequencies of every clock can be known to the receiver, synchronization can be kept for the rest of the time.

After N packets from a source have been received the reference time for this source is estimated. If the process of estimating the reference time is repeated 2 k times, the average value of t_{ref} is evaluated for two intervals, one from 0 to the k'th run and another from the k+1'th to the 2k'th run. As a result, the error of the estimated reference time to the true reference time for these two intervals will be reduced by $\sqrt{k}$. A justification for this is given in Appendix D.

Since what we do is an averaging process over kN packets, the obtained value of the reference time is true for the two time instances kNT/2 and 3kNT/2. These are the middle points of the two intervals [0,kNT] and [kNT+T,2kNT] respectively. The reason is that only in these two time instances the accumulated contribution of the time offsets in estimating the reference time is zero. More about the validity of the above statement is given in Appendix E.

Hence, the actual reference time will equal the measured reference time $\hat{t}^i_{ref,n}$ ± an error less than $\frac{\epsilon}{\sqrt{k}}$, i.e. $\left|t^i_{ref,n} - \hat{t}^i_{ref,n}\right| \leq \frac{\epsilon}{\sqrt{k}}$, where n is either kN/2 or 3kN/2. Therefore, the maximum error we are making in estimating the time offset during the time interval [kNT/2 , 3kNT/2] is

$$\left|\left(T^i_{off}|_{t=kNT}\right) - \hat{\Delta}T^i\right| \leq \frac{2\epsilon}{\sqrt{k}}\,, \tag{6.4.12}$$

where $\hat{\Delta}T^i = \hat{t}^i_{ref,\frac{3kN}{2}} - \hat{t}^i_{ref,\frac{kN}{2}}$ is the estimated time offset and $T^i_{off}|_{t=kNT} = t^i_{ref,\frac{3kN}{2}} - t^i_{ref,\frac{kN}{2}} = \left(f^i - f^R\right)kNT$ is the actual time offset. The estimated frequency

offsets $\widehat{\Delta F}^i$ of each source i can now be determined as

$$f^i - f^R = \frac{T^i_{off}|_{t=kNT}}{kNT} \Rightarrow \widehat{\Delta F}^i = \frac{\widehat{\Delta T}^i}{kNT}. \tag{6.4.13}$$

Our objective is to find the future time instance ΛNT that the error will reach the design specification error ϵ . Imagine a straight line drawn from the time instance kNT/2 to the time instance 3kNT/2. At these two instances, the maximum error we are making in estimating the reference time $\widehat{t}_{ref,n}$ is less than $\pm\frac{\epsilon}{\sqrt{k}}$. Therefore, the maximum tangent this line can have compared to the line that introduces no error is $\left(\frac{2\epsilon}{\sqrt{k}}\right)/kNT$ as illustrated in Figure 6.4.5. Observe that the real meaning of the slope is the maximum error we are making in estimating the frequency offset, i.e.

$$\left|\left(f^i - f^R\right) - \widehat{\Delta F}^i\right| = \frac{\left|\left(T^i_{off}|_{t=kNT}\right) - \widehat{\Delta T}^i\right|}{kNT} \leq \frac{\frac{2\epsilon}{\sqrt{k}}}{kNT}. \tag{6.4.14}$$

An important note here. The slope derived from Figure 5 is independent to the individual frequency offsets of the clocks $f^i - f^R$. A frequency offset for a specific source i might be 10 times larger than the frequency offset for source j but the absolute error in determining the frequency offsets will be the *same* for both sources.

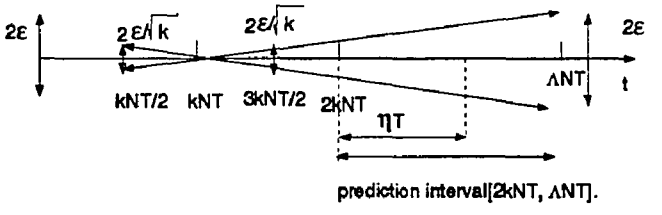

Figure 6.4.5 Schematic Diagram of the Prediction Algorithm.

To compute Λ we first note that at time 3kNT/2 we might start with as high an error in our estimation as $\pm\frac{\epsilon}{\sqrt{k}}$ and second, we start predicting only after kN/2 packets have

been received. Therefore, the following equation holds

$$\frac{\epsilon}{\sqrt{k}} + \left(\frac{\frac{2\epsilon}{\sqrt{k}}}{kNT}\right)\left(\Lambda NT - \frac{3kNT}{2}\right) \leq \epsilon \qquad (6.4.15)$$

For meaningful value of k, such that $\Lambda > 2k$, solving (6.4.15) we get $\Lambda = \frac{\sqrt{k}+2}{2}k$.

In Figure 6.4.6 we present a simple example in order to clarify the involved implementation of the prediction algorithm presented in this section. We assume that the receiver accepts packets from 3 sources. Suppose that at times (1/2)kNT and (3/2)kNT the reference times of each source are as shown in Figure 6.4.6. Specifically, let at time (3/2)kNT the distances of each reference time to the boundary between reference interval *n* and reference interval *n+1* be: $\delta_1 T$ for source 1, $(1-\delta_2)T$ for source 2, and $\delta_3 T$ for source 3. The time offsets $\widehat{\Delta}T^i$'s, which represent the amount of time the reference time of each source *i* has moved during the time interval [(1/2)kNT, (3/2)kNT], are indicated as $\widehat{\Delta}T^i$ in the figure. Using equation (6.4.13), the estimated frequency offsets $\widehat{\Delta}F^i$'s are determined. Some of the clocks at the sources (sources 1 and 3 in the example) will run faster than the receiver clock, whereas the rest of them (source 2) will run slower. The time interval $\eta_i T$ taken from time 2kNT as shown in Figure 6.4.5, where at the end of it the reference time from source *i* will move out of the current reference interval, can now be determined as the solution of the following equation

$$\delta_i T = \left|\widehat{\Delta}F^i\right|\left(\frac{kNT}{2} + \eta_i T\right) \Rightarrow \eta_i = \frac{\delta_i}{\left|\widehat{\Delta}F^i\right|} - \frac{kN}{2}, \quad 0 \leq \eta_i \leq (\Lambda - 2k)N \ . \qquad (6.4.16)$$

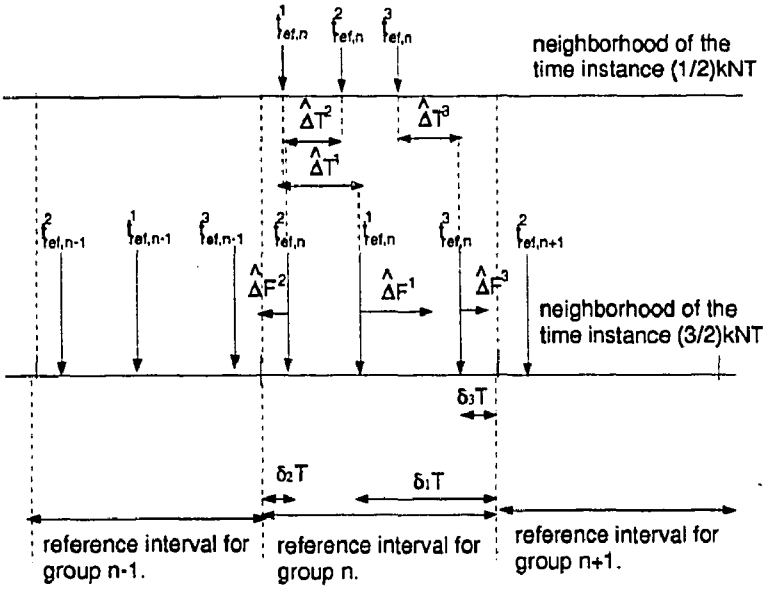

Figure 6.4.6 Example in calculating the future times when reference times move out of their reference interval T.

Finally, we see that the minimum waiting time for playback after the execution of the prediction algorithm will not have the term T_{off}^{i} as already noted in section 2.3 and Equation (6.3.4.9) becomes

$$W_{min} = T + \max_{i} \left\{ \hat{\Delta}_{max}^{i} \right\} + \epsilon \, , \; 1 \leq i \leq M \, . \tag{6.4.17}$$

By employing *feedback* the term T caused by the phase offsets can be reduced leading to minimum waiting time for playback. The issue of feedback will be discussed in the next section.

6.4.1 Performance Analysis of the Frequency Offset Estimation Method

In this subsection we carry out a performance analysis of the Frequency Offset estimation methodology we have used as described in Figure 6.4.5 and Equations (6.4.12) through (6.4.14). Specifically, after the mean square error is determined in Appendix F, we show that our method produces nearly optimum results. It is nearly optimum in the sense that the mean square error is only about 25% higher than the minimum mean square error determined by the Cramer-Rao bound.

The expression for the mean square error for the case of only N packets received was shown in Appendix F to be

$$E(\epsilon^2) = \frac{16\sigma_\Delta^2}{N^3} . \qquad (6.4.1.18)$$

In the paper by J. Wolf and J. Schwartz[60] using the Cramer-Rao bound (see also [61]), they show that the best estimation of the frequency offset one can have will yield a minimum mean square error of

$$E(\epsilon^2)_{best} = \frac{12\sigma_\Delta^2}{N(N+1)(N+2)} . \qquad (6.4.1.19)$$

The validity of Equation (6.4.1.18) is also demonstrated with simulations as shown in Figure 6.4.7. In Figure 6.4.7 we have plotted both the theoretical results from Equation (6.4.1.18) and simulations for two cases: a) for the case that the *p.d.f.* of the Δ_n^i's is Normal with zero mean and variance one, and b) for the case that the *p.d.f.* of the Δ_n^i's is uniform with zero mean and variance one. In addition, a plot of the optimum mean square error is included for comparison purposes. Our method provides a computationally cost effective (actually, 2k+2 additions and 3 divisions) means to approach very close to optimum results. To attempt to get any closer to the optimum solution would require much additional calculations (related to evaluation of the pseudo-inverse matrix) and would be impractical.

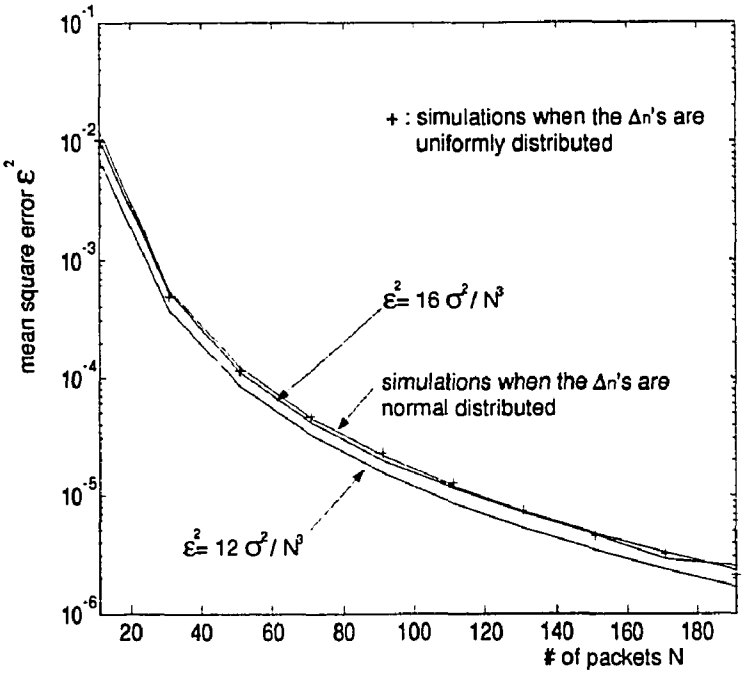

Figure 6.4.7 The mean square error of the frequency estimation method.

6.5 Feedback

It is obvious that the minimum waiting time obtained with Equation (6.4.17) is not the optimum. A source *i* with reference time at the beginning of the interval T will be delayed an additional time T in order for the packets from source *i* to be played back. Since T can be one or two times the maximum jitter experienced in the network, quality of service will be degraded due to excessive delay. While the second and the third terms in Equation (6.4.17) which have to do with the maximum jitter and the error cannot be minimized further, the first term T can be minimized if feedback is employed. As a reminder, T has to do with the phase offsets existing among the different participants of a conference. We will distinguish two cases depending on whether an estimation of the frequency offset is available. One from time NT until the time 2kNT, and the other from time 2kNT to the end of the conference.

Since the estimation of the reference times is done every NT seconds, a feedback packet from the receiver to source i need to be sent every NT seconds. There will be a delay from the moment the feedback packet is sent to the moment it arrives at the source and takes effect. The packets that arrive at the receiver during this delay cannot be used towards the estimation of the next reference time.

6.5.1 Feedback in the Time Interval [NT,2kNT].

During this period of time [NT,2kNT], we do not have an estimation of the frequency offsets and it is impossible to predict the exact time offsets acquired during the time interval NT between the clock at the receiver and the clock at the sender. Nevertheless, as we have done in Equation (6.3.4.9), we assume that in a time interval NT the maximum possible time offset two clocks in the network can experience is $\left|(T_{off}|_{t=NT})\right|$. To compare the reference times of the different sources, we arbitrarily choose to measure their time distances from the midpoint M of their current reference interval. Thus, when the reference times are estimated, their distances d^i to point M will be computed as well. A distance will be called positive when the reference time of the source is at the left to the point M, and negative when it is at the right. If the feedback packet carries the value d^i to each source i, and the source i readjusts itself properly, the reference time for source i for the next NT seconds will be very close to point M, and actually will be within the interval $\left[M - \left|(T_{off}|_{t=NT})\right|, M + \left|(T_{off}|_{t=NT})\right|\right]$. Source i will either pause time d^i if d^i is positive, or precede the following data of time duration (T — d^i) with noisy data of duration d^i, if d^i is negative. The minimum waiting time will now be

$$W_{min} = 2\left|(T_{off}|_{t=NT})\right| + \max_i \left\{\Delta^i_{max}\right\} + \epsilon\ ,\ 1 \le i \le M\ . \qquad (6.5.1.20)$$

The factor 2 in front of the time offset term is necessary because we do not know the direction of the maximum frequency offset that may exist in the network.

6.5.2 Feedback from Time 2kNT to the End of the Conference.

From the moment the prediction algorithm is in effect, accurate estimates of the frequency offsets between the clocks at the sources and the clock at the receiver are obtained. Let's assume that the size of the maximum frequency offset occurs for source j . In NT seconds, this will lead to a maximum time offset $T^j_{off}|_{t=NT}$. Let d^j be the distance of the reference time for source j to the midpoint M of the reference interval. The feedback packet brings this information to source j and source j readjusts itself so that its new reference time is at point M. After time NT the reference time for source j will move to a new point, point Z. We have $|M - Z| = \left|\left(T^j_{off}|_{t=NT}\right)\right|$. Point Z will be to the left of point M if the time offset is negative, i.e. the clock at the source j is slower than the clock at the receiver, and to the right of point M if it is positive, i.e. the clock at the source j is faster than the clock at the receiver. Without loss of generality, let us assume that the clock for source j is faster than that of the receiver and consequently point Z will be at the right of point M as shown in Figure 6.5.8. Then we compute the distances for the rest of the sources the following : a) for the sources with positive time offsets (source 1 in the figure), the distances will be computed from their reference times to point M. b) for the sources with negative time offsets (source 2 in the figure), the distances will be computed from their reference times to point Z. The distances defined in this way will be sent back with the feedback packets to the corresponding source, i.e. distance d^i to source i. When the sources receive the feedback packets, they will readjust the reference times to coincide with either point M or point Z. Since we have used the maximum possible clock offset of any of the sources, the reference times are guaranteed to be within the interval $[M,Z]$ during the next time interval NT. The minimum waiting

time will now be

$$W_{min} = \left| \left(T^{j}_{off} |_{t=NT} \right) \right| + \max_{i} \left\{ \hat{\Delta}^{i}_{max} \right\} + \epsilon \, , \; 1 \leq i \leq M \, . \qquad (6.5.2.21)$$

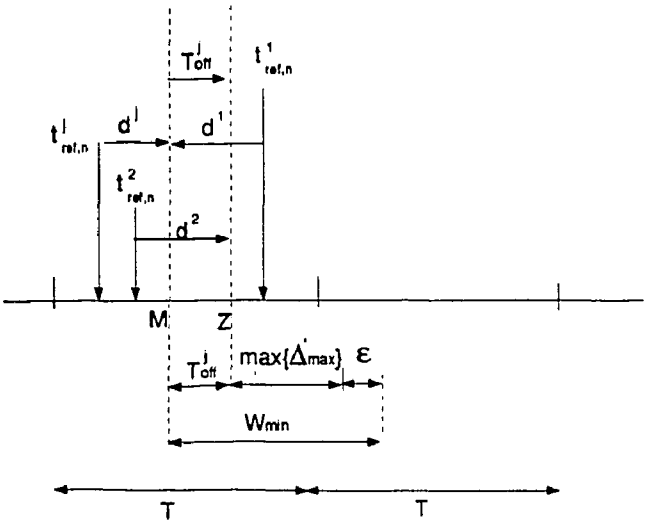

Figure 6.5.8 Determination of the distances d^i the reference time of each source i has to be readjusted after the reception of a feedback packet.

A few words about Equation (6.5.2.21) are in order. The minimum waiting time defined by Equation (6.5.2.21) gives *almost the same minimum* waiting time with a case of a multimedia conference with two participants, i.e. a sender and a receiver. The minimum waiting time is the same with the point-to-point conference in the specific case that the sender exhibits the maximum frequency offset and the maximum jitter. The same comments are also applied for the minimum delay time. The reason is that our algorithm does discriminate packets arriving from sources with different average time delay, and packets from the same source suffer only the average time delay calculated for their own source. The maximum delay for the case of only two participants will be

$$D^{i}_{T} = \bar{D}^{i} + \left| \left(T^{i}_{off} |_{t=NT} \right) \right| + \hat{\Delta}^{i}_{max} + \epsilon \, , \qquad (6.5.2.22)$$

where $\bar{D}^i$ is the average delay packets generated from sender i experience to reach the receiver. In the presence of other sources, the maximum delay packets from source i experience, is

$$D_T^i = \bar{D}^i + W_{min} = \bar{D}^i + \left|\left(T_{off}^j|_{t=NT}\right)\right| + \max_i \left\{\hat{\Delta}_{max}^i\right\} + \epsilon \ , \ 1 \le i \le M \ . \quad (6.5.2.23)$$

We see that the overhead to handle the case of M participants in comparison to the point-to-point conference is negligible, and therefore our algorithm is shown to provide a solution very close to the optimum in determining the minimum waiting/delay time.

6.6 Performance Bounds in Reference Time Estimation.

6.6.1 Estimation Error in the Reference Time

The error in estimating the reference time will come from the finite number of packets N received at the receiver. As it is shown in [2, 7, 3, 59] , the probability density function of packet arrivals is approximately normal or Rayleigh (The probability density function of packet arrivals are shown to be approximately normal when the message size is relatively small, while it tends to be Rayleigh when the message size is large (~24Kbytes)[2]) and $\Delta^i{}_{min} \le \Delta^i{}_{max}$.

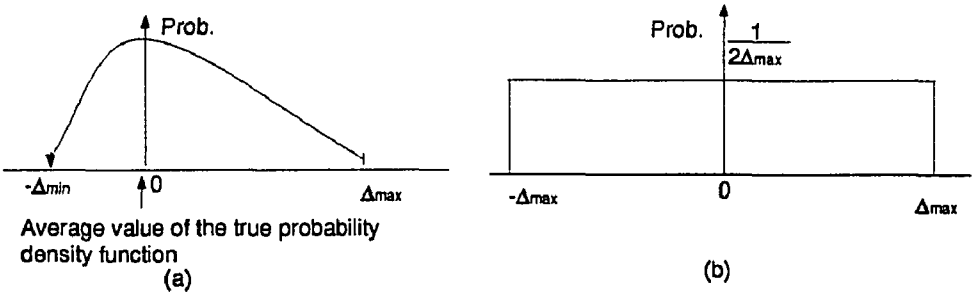

Figure 6.6.9 (a) Example of a real probability density function. (b). The worst case scenario. Uniform density in the interval $[-\Delta_{max}, \Delta_{max}]$.

We will derive an upper bound in the probability of error of our estimation. In Figure 6.6.9 (a), a bell shaped probability density function is depicted to model the distribution of real packet arrival times. Consequently, using this model is evident that a probability density function of realistic packet arrival times has smaller variance than a uniform density function in the interval $[-\Delta_{max}, \Delta_{max}]$. Its average value is the same average value estimated for the actual density and its variance is $\sigma_u^2 = \frac{\Delta_{max}^2}{3}$. A loose upper bound for the probability of error is obtained by Chebychev 's inequality which is solely based on the variance of the probability density function. For this reason, density functions with larger variances lead to larger bounds (Upper bound in the probability of error, not in the probability of error itself. Furthermore, this upper bound is derived using the Chebyshev inequality and not the Chernoff bound. But as we will see in Figure 6.6.10 (b), the uniform case leads to a larger upper bound for at least one case of the Rayleigh function, the exponential and the normal) in the probability of error. Therefore, the assumption of a uniform density function of the packet arrivals as shown in Figure 6.6.9 (b) will lead to an upper bound in the estimation error of the reference time than in the case of a realistic probability density function of packet arrivals.

As an example, let us assume that real packet arrivals follow a normal density function. The same procedure can be followed for an example where the packet arrivals follow the Rayleigh probability density function as well as any other bell-shaped density function. Using the Chernoff bound, if N packets are received and assuming statistical independence in the arrival time of the packets, the probability that the average value

lies outside a confidence interval $\frac{\epsilon}{\Delta^i_{max}}$ will be (see [62])

$$P\left(\frac{\varepsilon}{\Delta^i_{max}}\right) = P\left[\left|t^i_{ref,n} - \hat{t}^i_{ref,n}\right| \geq \frac{\varepsilon}{\Delta^i_{max}}\right] = 2P\left[\frac{1}{N}\sum_{n=1}^{N}\Delta^i_n \geq \frac{\varepsilon}{\Delta^i_{max}}\right]$$

$$\leq 2\left[E\left\{e^{\lambda_0\left(\Delta^i_n - \frac{\epsilon}{\Delta^i_{max}}\right)}\right\}\right]^N \Rightarrow \frac{1}{2}P\left(\frac{\varepsilon}{\Delta^i_{max}}\right) \leq$$

$$\begin{cases} \left[E_N\left\{e^{\lambda_0\left(\Delta^i_n - \frac{\epsilon}{\Delta^i_{max}}\right)}\right\}\right]^N = e^{-\lambda_0\frac{\epsilon}{\Delta^i_{max}}N}\left(e^{\lambda_o^2\sigma_\Delta^2 N/2}\right) \\ \left[E_U\left\{\epsilon^{\lambda_0\left(\Delta^i_n - \frac{\epsilon}{\Delta^i_{max}}\right)}\right\}\right]^N = e^{-\lambda_0\frac{\epsilon}{\Delta^i_{max}}N}\left[\frac{sinh(\lambda_0\Delta^i_{max})}{\lambda_0\Delta^i_{max}}\right]^N, \ \sigma^2_\Delta = \frac{(\Delta^i_{max})^2}{3} \end{cases} \tag{6.6.1.24}$$

where λ_0 is given as the solution (With the help of tables of integrals given in pages 307, 338 [63].) of the following equation

$$\frac{E\left[\Delta^i_n e^{\lambda_0\Delta^i_n}\right]}{E\left[e^{\lambda_0\Delta^i_n}\right]} = \frac{\varepsilon}{\Delta^i_{max}} \Rightarrow \lambda_o = \begin{cases} \frac{\frac{\varepsilon}{\Delta^i_{max}}}{\sigma^2_\Delta}, & Normal \\ \frac{\tanh\left(\lambda_0\Delta^i_{max}\right)}{\Delta^i_{max} - \frac{\varepsilon}{\Delta^i_{max}}\tanh\left(\lambda_0\Delta^i_{max}\right)}, & Uniform \end{cases} \tag{6.6.1.25}$$

and E_N, E_U are the expectation operators under the Normal and Uniform density functions respectively. $P\left(\frac{\varepsilon}{\Delta^i_{max}}\right)$ signifies the probability that the estimated reference time differs from the actual reference time more than $\frac{\varepsilon}{\Delta^i_{max}}$. Equation (6.6.1.25) for uniform probability density function is to be solved numerically, and for the following values of interest 0.05, 0.1 and 0.2 of error $\frac{\epsilon}{\Delta^i_{max}}$, the solutions of λ_0 are 0.15 , 0.3 and 0.6 respectively. Δ_{max} is taken as 1 in all cases.

For comparison purposes between the normal, the uniform and the Rayleigh density functions, we chose the same $\Delta^i_{max} = 1$. The variance is chosen $\sigma^2_\Delta = 0.2$ such that both the sum of the probabilities outside $\Delta^i_{max} = 1$ in the normal function is negligible and the variance has the same value as that for the Rayleigh density function with its probability of error plotted in Figure 6.6.10 (b). Then, ϵ will be a fraction of $\Delta^i_{max} = 1$. Therefore,

Equation (6.6.1.25) for normal probability density function and for the following values of interest 0.05, 0.1 and 0.15 of error $\frac{\varepsilon}{\Delta^i_{max}} = \varepsilon$, the solutions of λ_0 are 0.25 , 0.5 and 0.75 respectively.

6.6.2 Numerical Examples

The proposed interparticipant synchronization algorithm introduced in this paper needs to be executed every *NT* seconds. So far, we have not provided the reader with a specific value of *N* because *N* will change according to the application and the network environment. However, using the theory developed in the previous section, we provide explicitly the trade-offs among the three parameters: Probability of error, Confidence interval $\frac{\varepsilon}{\Delta^i_{max}}$, and the number of packets *N*. Depending on the application (how accurate we really need to be), and on the network environment (how large the frequency drifts of the clocks and the network jitter are), the designer can choose the value of *N* suited for his application.

A realistic example will be the following. Assume workstations with voice and video capabilities. Since the speed of the video boards is 30 frames per second, the transmission period of the voice packets is also assumed to be 30 packets per second or 1 packet per 33ms (=*T*). In our lab, the pair of workstations where the video-boards are attached have a difference in clock rates which cause a timeshift of about 1 second per half an hour or $\widehat{\Delta}F^i = (30 \times 60sec)^{-1}$. The maximum value of jitter which we observed for the case of UDP is about 10ms (=Δ_{max}). If the tolerance desired is less than 3ms and the error is distributed 1 ms for the time offset and 2 ms for the estimation error, then by allowing 1 ms of timeshift in the interval *NT*, *NT* must equal $NT = \frac{\widehat{\Delta}T^i}{\widehat{\Delta}F^i} = \frac{1ms}{(30\times60sec)^{-1}} = 1.8sec.$ This means that N is 54 (1.8×30). With N = 54, as it will be shown later in this section, the estimation error is much less than 2 ms

(or $0.2\Delta_{max} = 0.2 \times 10ms$) for probability of error $P\left(\frac{\epsilon}{\Delta'_{max}}\right) = 0.05$.

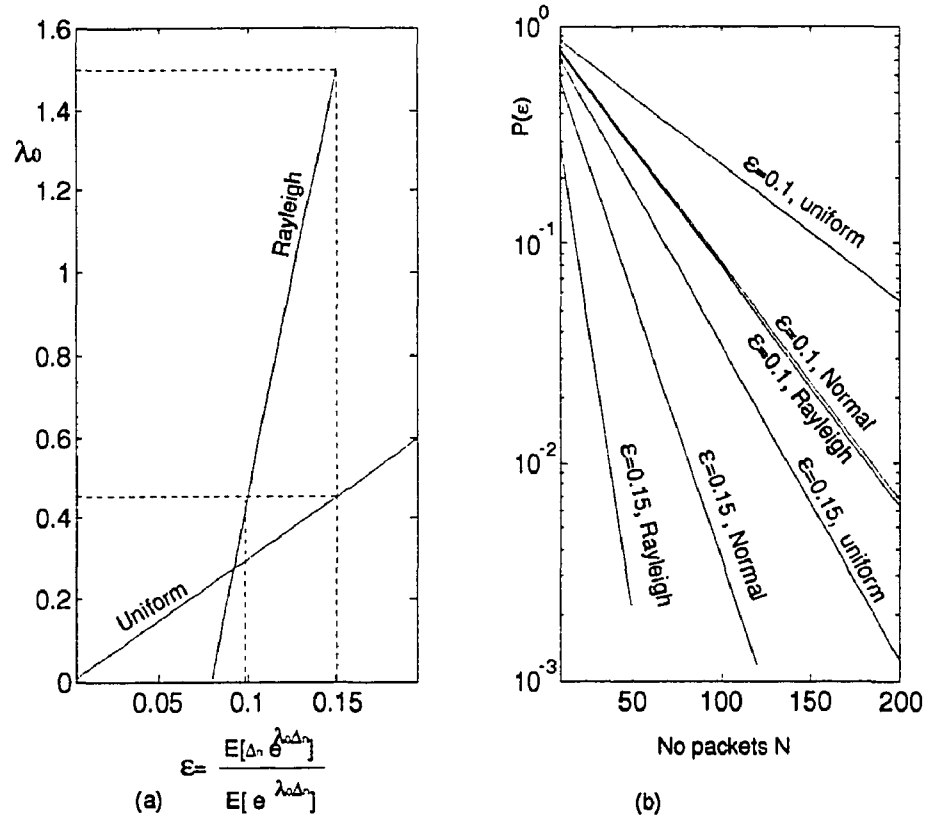

Figure 6.6.10 (a) Numerical evaluation of λ_0 for different range of ϵ $\left(= \frac{\epsilon}{\Delta'_{max}}\right)$. (b) Numerical evaluation of the probability of error P(ϵ) derived from the Chernoff bound using the λ_0's from (a) for the Uniform and the Rayleigh density. The λ_0's for the Normal density were derived theoretically before and are 0.5 and 0.75 for ϵ= 0.1 and 0.15 respectively.

The values of λ_0 are also derived numerically for the Rayleigh density with parameter b = 0.1 and for the uniform density. These are plotted versus the error ϵ $\left(= \frac{\epsilon}{\Delta'_{max}}\right)$ in Figure 6.6.10 (a). In Figure 6.6.10 (b) we have plotted the theoretical upper bounds of the probability of error P(ϵ) derived from the Chernoff bound for the uniform, normal and Rayleigh density functions. As it is shown in Figure 6.6.10 (b), with ϵ=0.1, P(ϵ) = 0.05 can be achieved with N=120 when using the Rayleigh or the normal, and with N =

200 when using the uniform. However, we notice a dramatic decrease in the probability of error when the confidence interval is allowed to be ϵ=0.15. P(ϵ) = 0.05 can now be achieved with only 25 packets using the Rayleigh density function. This is very important result especially for the *connection or setup time* for a conference. During this period, the received packets are considered to be used only for conference management purposes, such as synchronization, bandwidth allocation, etc.

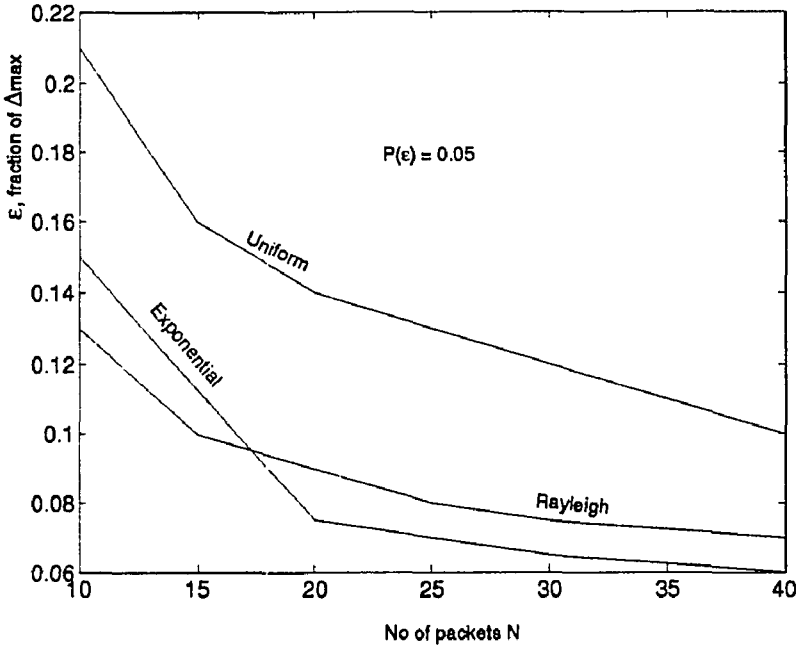

Figure 6.6.11 Simulation results for three density functions: uniform, exponential and Rayleigh with parameter b=0.1. Confidence interval $\frac{\epsilon}{\Delta'_{max}}$ is plotted vs the number of packets N.

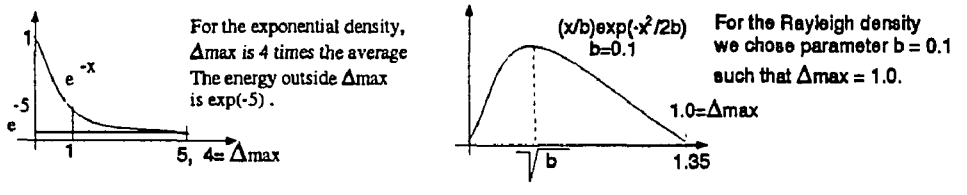

Figure 6.6.12 Simulation models for exponential and Rayleigh density functions used in Figure 6.6.11.

Notice here that Equation (6.6.1.24) only shows the upper bounds. In order to get a better estimate on the relation between the error ϵ and N for constant probability of error, we resort to simulations. In Figure 6.6.11 we present simulations for three different density functions: uniform, exponential and Rayleigh. The simulation models for the probability density functions of exponential and Rayleigh functions are shown in Figure 6.6.12. For constant probability of error $P(\epsilon) = 0.05$, the confidence interval ϵ (which is presented here as fraction of Δ_{max}) is plotted versus the number of packets N received. It is clearly seen from the figure that the uniform case is the worst from the other two cases. Nevertheless, the results we get from simulations are much better than the ones using the Chernoff bound. Specifically for the Rayleigh case, confidence interval $\epsilon = 0.1\Delta_{max}$ can now be achieved with only 15 packets and not 120 as when using the Chernoff bound for the same probability of error 0.05. If the transmission period of voice packets is 33ms (or 30 packets per second), then a workable synchronization can be achieved within 0.5s.

6.7 Concluding Remarks for Chapter 5.

In this chapter we attacked the problem of synchronization among different computers participating in a multimedia conference. More specifically we were concerned with the synchronization of the packets/messages at the application level.

The core of the proposed *Interparticipant Synchronization Algorithm* is the concept and the estimation of the reference times. Once the reference times are estimated, the maximum jitter and the minimum waiting time can be determined, and consequently, synchronization is achieved for the next N packets. We then have expanded the synchronization algorithm by presenting a methodology of predicting the time/frequency offsets. Both the accuracy and the computational complexity of the algorithm are improved further by the prediction algorithm.

Time stamping or sequence numbering associated with packets at the application level allows our algorithm to work even with unreliable transport mechanisms such as *UDP*. The synchronization algorithm is also shown to encompass cases where traffic sources transmit packets with different periods. The degree of accuracy of synchronization depends on how accurate the estimation of the reference times are. Modeling the probability density function of packet arrivals with a Rayleigh, Normal and Uniform density functions, an upper bound in probability of error derived from Chernoff bound is plotted with respect to the number of received packets N. For instance, by allowing an error of 0.15 of Δ_{max}, only 25 packets are required for $P(\epsilon) = 0.05$ using the Rayleigh model. Using simulations, the same performance can be achieved with only 15 packets. This means that at the connection/setup time, workable synchronization can be achieved in a very short time, in the neighborhood of one second. This displays the high feasibility of the proposed algorithm.

By employing feedback, we are led to the optimum waiting time and consequently to the minimum delay that can be achieved for each source, optimum in the sense that the minimum waiting time obtained with our algorithm is the same time one would have obtained if he only had considered only two participants, i.e. one sender and one receiver.

Chapter 7 Efficient Estimation of the Frequency Offset in Real-Time Network Applications

7.1 Motivation

The number of computer networks has shown an exponential increase during the past two decades. While computer networks have been used mainly as a vehicle for transferring data so far, due to a vast growth in many aspects of computer related technology transfer of real-time voice and video through computer networks is expected to become a reality soon. However, transfer of real-time of data, among other things, imposes strict requirements on time delay and synchronization. The last requirement, due to the packet switching technology employed in computer networks, has led many researchers in developing algorithms / protocols for Network Timing Synchronization [20, 21, 22, 23, 24]. Nonetheless, the nature of Network Timing Synchronization leads to complicated algorithms that make a possible implementation of such an algorithm / protocol questionable by many researchers on the field. An alternate approach to Network Timing Synchronization is proposed in this Chapter. Clock (Timing) Synchronization is not anymore the responsibility of the network itself but that of the participants communicating in real-time using the computer network.

Network Timing Synchronization provided to a group of hosts only when they are communicating with each other alleviates many of the burdens existed for a "global" or "regional" network timing synchronization. Much of the complexity of such an approach arises from the requirement that good operation is needed also in the presence of a node failure. This is especially true for a point-to-point communication where a node failure will directly be translated as the end of the connection. Another requirement adding to the complexity of a "global" network timing synchronization protocol is that such a

protocol tries to provide the network with a global or universal clock, in the sense that the clocks of each host try to show the same time at any time. However, with the concept of the reference times or expected arrival times that we have introduced in [33, 32] and expanded in [53, 54], a universal time is no longer necessary for real-time applications. This is due to the concept of the reference times because now packets from different participants become synchronized not according to their generation times but according to their *expected arrival times*. Thus, only estimation of the frequency offset is needed and not of the actual time offset existing among the different hosts. Since different hosts are randomly geographically distributed propagation delays are different in each connection complicating any synchronization protocol trying to estimate the time offset.

7.2 Overview

In this Chapter, the issue of the estimation of the frequency offset between sender and receiver in a packet switching network environment in real-time applications is discussed in detail. The present Chapter provides the receiver with a mechanism to estimate the generation period of the packets at the sender $\hat{T}^s$. knowledge of $\hat{T}^s$ both lowers the frequency of the execution and improves the synchronization of the algorithm presented in [53, 54]. The frequency offset estimation is based on evaluating the tangent among two subsets of the arrived packets which are constructed so they lead to minimum mean square error. The estimation is further improved by combining the different estimations done locally at each host. The coupling of the estimations from different hosts provides adequate estimation to the frequency offset also to the participant who has not received any packets, or very few, during the time interval in consideration. The computational cost of the proposed algorithm is significantly lower than the least squares approach, while its mean square error is only a fraction (in the simulation example only 1%) higher

than the optimum. All theoretical results are validated using simulations, and the packet arrival times are assumed to be independent.

We would like to emphasize that the operation of the proposed algorithm is based solely on the packets exchanged between the two users and no extra packets to be transmitted are required. The frequency offset is assumed to be a linear function of time. In reality, as denoted by Mitra in [20] and Mills in [21], for time intervals in the order of micro-seconds the frequency offset is a non-linear function of time. However, for real-time network applications the inter-arrival times of the packets is in the order of tens of milli-seconds resulting in cancellation of the non-linearities. The model of the traffic used is *noncontinuous periodic*. This model is well suited for the transmission of voice packets since a conversation exhibits a pattern of silence-talkspurt intervals. An illustrative example of how the voice packets are transmitted is shown in Figure 4.2.1. The shaded areas in the figure denote talkspurt intervals while the rest are silent. Even though a talkspurt interval begins or ends at a time that does not coincide with the phase of the period T, packets are generated only *in-phase* (the vertical lines indicated in the figure) with each other. For example, although the time interval between the vertical lines numbered 2 and 3 in the figure is only partially filled with voice data, a whole packet will be sent starting from the vertical line numbered 2. In addition, a sequence number representing the sequence number of the time period the packet was generated at the sender will be attached to the packet. As a result, the packet generated at the time period (slot) between the vertical lines 2 and 3 will be assigned the sequence number 2.

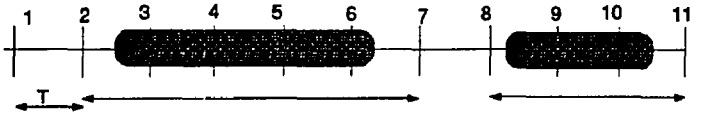

Figure 7.2.1 Representing the traffic as noncontinuous periodic.

In section 3, the definitions of the network jitter and of the reference times are given. In section 4 our algorithm for estimating the frequency offset and its performance analysis with respect to the mean square error as criterion is presented. In section 5, a theory related to the coupling of two frequency offset estimations is developed and simulations which validate the theory are presented. In section 6, the optimum way to divide a set of N packets so that when our algorithm is applied to lead to the minimum mean square error is presented. Finally, in section 7 we present our concluding remarks for this Chapter.

7.3 Definitions of the Network Jitter and Reference Times

The same definition of the network jitter and the conceptual framework developed in our earlier paper [33, 32] about interparticipant synchronization are also used in the present paper. The definitions of the network jitter and the reference times are as follows. As shown in Figure 4.3.2, the sender periodically transmits packets with *nominal* period T. However, due to frequency mismatches between the clock at the sender and the clock at the receiver, the actual transmission period of the sender as seen with the receiver's clock will be T^s. The average delay time packets experience to arrive at the receiver from the sender is denoted as D. If there were no variation in the delay and each packet were experiencing a constant delay D, then packets would be arriving at points indicated in the graph as $t_{ref,n}$. These points are called the reference times and they satisfy the following condition

$$\lim_{N\to\infty} \sum_{n=1}^{N} (t_n - t_{ref,n}) \to 0 \quad . \tag{7.3.1}$$

If the exact location of the arrival time of one packet, let us say packet n_1, which experienced exactly the average delay D could be found, then the reference time of any other packet n_2 is given by the following expression

$$t_{ref,n_2} = t_{ref,n_1} + (n_2 - n_1)T^s \quad . \tag{7.3.2}$$

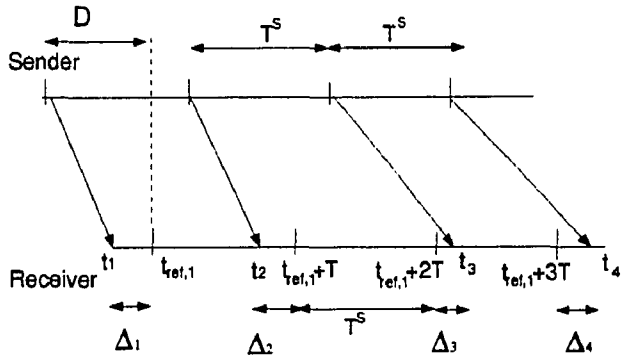

Figure 7.3.2 The definition of the network jitter.

If the actual packet arrivals are denoted as t_n, then the network jitter each packet experiences is defined as the difference of the actual arrival time minus the corresponding reference time

$$\Delta_n = t_n - t_{ref,n} \quad . \tag{7.3.3}$$

In the following sections, the time X where the time offset contributions of all the packets cancel out will be frequently used . Let $\mathfrak{N}$ be the set of all the packets arrived at the receiver, $\mathfrak{N}_E$ be a subset of $\mathfrak{N}$ such as $\forall n \in \mathfrak{N}_E \Rightarrow t_n \leq X$, and $\mathfrak{N}_L$ be the complimentary subset of $\mathfrak{N}$ such as $\forall n \in \mathfrak{N}_L \Rightarrow t_n > X$. Then, the time X can be determined as follows

$$\sum_{(n\in\mathfrak{N}_E)} (X - t_n) = \sum_{(n\in\mathfrak{N}_L)} (t_n - X) \Rightarrow \; X = \frac{1}{N} \sum_{(n\in\mathfrak{N})} t_n \quad . \tag{7.3.4}$$

7.4 Frequency Offset Estimation In The Presence of Silent-Talkspurt Intervals.

There are two criteria that characterize the state between two clocks, time offset and frequency offset. Time offset is the time difference that exists between the two computers at the time of question, while frequency offset is the rate of change of the time offset. Frequency offset is most bothersome because even if at one point in time clock synchronization is achieved, frequency drift would drive the clocks out of sync as time progresses. This shows the significance of having accurate estimation of the frequency offset because then continuous synchronization could be achieved once synchronization was achieved for a point in time.

In our previous work [33, 32], we proposed a methodology in estimating the frequency offset under the assumption that packets were generated periodically and *continuously* at the source. In this section our previous method is expanded to encompass the case of a source generating packets periodically but not continuously.

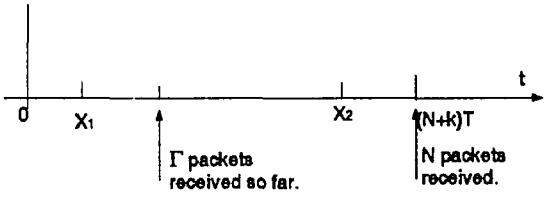

Figure 7.4.3 The N packets received are divided into two subsets with Γ and $N - \Gamma$ packets each.

In a time interval $(N + k)T$, N packets have been received, while k is the number of time slots T that the sender did not generate any packets. To estimate the frequency offset between the clocks at the sender and at the receiver, the N packets are divided into two subsets. In a later section, the optimum way to divide this set into two subsets is shown to satisfy the condition of Equation (7.6.15). Let us assume that the proper division into two sets has been done, and as shown in Figure 7.4.3, Γ is the number of packets of

the first set and $N-\Gamma$ is the number of packets of the second set. Each packet arriving at the receiver carries along the sequence number of the time slot which was generated at the sender. For the example shown in Figure 4.2.1, the first packet transmitted will carry the number 2, the second the number 3, and the sixth the number 8 since a silent interval of one period intervenes. The ordered set of the sequence numbers of all the arrived packets will be denoted as $\mathfrak{N}$, and the elements will be ordered in sequence from small to high according to their values. This implies that in a situation where a packet arrived out of sequence, the entry in the set $\mathfrak{N}$ will be corrected according to the ordering rule. The sequence numbers associated with the first Γ packets of the ordered set $\mathfrak{N}$ will constitute a subset which we will denote it as $\mathfrak{N}_{\Gamma}$, while the last $N-\Gamma$ packets will constitute another subset denoted as $\mathfrak{N}_{N-\Gamma}$.

Next, estimation of the points $\widehat{X}_1, \widehat{X}_2$ where the time offset contributions of the packets of the two sets cancel out are found using Equation (7.3.4). We have,

$$\widehat{X}_1 = \frac{1}{\Gamma} \sum_{n \in \mathfrak{N}_{\Gamma}} t_n \ , \ \widehat{X}_2 = \frac{1}{N-\Gamma} \sum_{n \in \mathfrak{N}_{N-\Gamma}} t_n \ . \tag{7.4.5}$$

The way points X_1, X_2 were constructed, any uncertainty in estimating them is due to the finite number of packets Γ and $N-\Gamma$ used for their estimations, and no component due to the frequency offset exists. Nevertheless, these two points carry with them the inherit periodicity T^s according to the clock at the sender since they are a function of the packet arrival times. This becomes more clear if Equation (6.2.3) is used and each packet arrival time t_n is expressed as follows

$$t_n = t_{ref,0} + nT^s - \Delta_n \tag{7.4.6}$$

Then, without loss of generality we can set $t_{ref,0} = 0$, and the expressions for $\widehat{X}_1, \widehat{X}_2$

become

$$\widehat{X}_1 = \frac{1}{\Gamma}\sum_{n\in\mathfrak{N}_\Gamma} nT^s - \frac{1}{\Gamma}\sum_{n\in\mathfrak{N}_\Gamma}\Delta_n \text{ and } \widehat{X}_2 = \frac{1}{N-\Gamma}\sum_{n\in\mathfrak{N}_{N-\Gamma}} nT^s - \frac{1}{N-\Gamma}\sum_{n\in\mathfrak{N}_{N-\Gamma}}\Delta_n \tag{7.4.7}$$

where $-\frac{1}{\Gamma}\sum_{n\in\mathfrak{N}_\Gamma}\Delta_n$ and $-\frac{1}{N-\Gamma}\sum_{n\in\mathfrak{N}_{N-\Gamma}}\Delta_n$ are the errors in estimating the exact location of X_1, X_2 with $\widehat{X}_1, \widehat{X}_2$. At the receiver side, the reference times assigned to the arriving packets represent time according to the clock at the receiver. The reason is that the reference time of each packet $t_{ref,n}$ will be represented as a function of T^R, i.e. $t_{ref,n} = t_{ref,0} + nT^R$, since T^s is unknown.

Let $\xi_2 = \frac{1}{N-\Gamma}\sum_{n\in\mathfrak{N}_{N-\Gamma}} n$ and $\xi_1 = \frac{1}{\Gamma}\sum_{n\in\mathfrak{N}_\Gamma} n$. Then the frequency offset is estimated as follows

$$\widehat{\Delta F} = \frac{\widehat{X}_2 - \widehat{X}_1}{(\xi_2 - \xi_1)T^R} \quad . \tag{7.4.8}$$

Using Equation (7.4.8) the estimated period of the sender is $\widehat{T}^s = \widehat{\Delta F} \times T^R$. The reference time $t_{ref,n}$ of any packet $n > N + k$ is computed at the receiver as

$$\widehat{t}_{ref,n} = \widehat{t}_{ref,N+k} + (n - N - k) \times \widehat{T}^s \quad . \tag{7.4.9}$$

The result displayed in Equation (7.4.9) plays an important role in the multimedia synchronization algorithm presented in [53, 54] by enabling the receiver to lower the frequency of the execution of the algorithm on one hand, and improving the synchronization on the other.

7.4.1 Performance Analysis of the Frequency Offset Estimation Method

In this section, a performance evaluation of our methodology in estimating the frequency offset is carried out. A comparison to the least squares approach is also presented that shows the near-optimal performance of our algorithm. Comparisons and

outlines of other algorithms dealing with the estimation of the frequency offset which is a linear function of time are presented in [60]. Suppose that a total of N packets are received during the time interval $(N+k)T$, where k is the number of time slots of duration T that the sender did not generate packets. Then these N packets are divided into two subsets with Γ and $N-\Gamma$ packets each. If the jitter Δ_n each packet experiences is independent and identically distributed (*i.i.d.*) of the other and its *p.d.f* has variance σ_Δ^2, then according to the Central Limit Theorem (see for instance [62]) the normalized sum of N such random variables has variance $\frac{\sigma_\Delta^2}{N}$. In our case, the random variables $y = \frac{1}{\Gamma} \sum_{n \in \mathfrak{N}_\Gamma} \Delta_n$ and $z = \frac{1}{N-\Gamma} \sum_{n \in \mathfrak{N}_{N-\Gamma}} \Delta_n$ have variances $\sigma_y^2 = \frac{\sigma_\Delta^2}{\Gamma}$, $\sigma_z^2 = \frac{\sigma_\Delta^2}{N-\Gamma}$, and means $\bar{y} = \bar{z} = 0$ (Note that y and $\bar{y}$ are different). The error in estimating the frequency offset is $\epsilon = \frac{(y-z)}{(\xi_2-\xi_1)T^R}$, and its mean square error is

$$E\left\{\left(\frac{y-z}{(\xi_2-\xi_1)T^R}\right)^2\right\} = \frac{E\left\{(y-z)^2\right\}}{(\xi_2-\xi_1)^2(T^R)^2} = \frac{\sigma_\Delta^2}{(\xi_2-\xi_1)^2(T^R)^2}\left(\frac{1}{\Gamma}+\frac{1}{N-\Gamma}\right). \tag{7.4.1.10}$$

The problem of the estimation of the frequency offset whose rate of change is a linear function of time is a well known problem and an optimal solution that gives the minimum mean square error already exists [64, 61]. However, such an approach (least squares, see p. 45 in [61]) will require the inverse of $H^T H$, where H is the observation matrix of the packet arrival times $H^T = \begin{pmatrix} 1 & 1 & \dots & 1 \\ t_1 & t_2 & \dots & t_N \end{pmatrix}$. The mean square error of the Least Squares approach is shown to be

$$MSE = \frac{\sigma_\Delta^2}{\sum_{n=1}^{N} (t_n - \bar{t})^2} \quad , \tag{7.4.1.11}$$

where $\bar{t} = \frac{1}{N} \sum_{n=1}^{N} t_n$. (The numbering of the packets here represents the sequence number a particular packet was transmitted by the sender). For the example in Figure 7.6.6,

the mean square error was computed according to Equation (7.4.1.11) and found to be 2.165×10^{-6}, a little lower than the mse of 2.37×10^{-6} achieved with our algorithm. The mse of the least squares method is also denoted in Figure 7.6.8 for comparison purposes.

7.5 Coupling of the Frequency Offset Estimations

When two individuals communicate on the phone, the average time both of them spend on talking is 90% of the time. But there is a difference when one considers the talkspurt interval of each human individually and different when one considers it as the aggregate talkspurt interval spent by both participants in the conversation. The reason is that for a large time interval a person may be speaking continuously while the other person is simply listening (monologue). Then, if the talkspurt time for a certain time interval is considered separately for each person, the person who speaks may have as high as 90% of a talkspurt interval, and the person who listens may have as low as 0%. In such a case, the estimation of the frequency offset will provide good results to the person who listens and poor to the person who talks. This situation has prompted us to consider a possible coupling (combination) of the two frequency offset estimations which are done independently at each communicating site.

After the local estimation of the frequency offset is done at each host, before applying it, the frequency offset estimation itself is forwarded to the remote host. Similarly, after some delay the frequency offset estimation done at the remote host should reach the local host. Having locally both estimations, the objective is finding the weighting coefficients w_1 and w_2 such that the resulting frequency offset estimation of the coupling system

$$\widehat{\Delta}F_{\texttt{coupling}} = w_1 \times \widehat{\Delta}F_{\texttt{local}} + w_2 \times \widehat{\Delta}F_{\texttt{remote}} \qquad (7.5.12)$$

gives the lowest mean square error. In addition, the weighting coefficients appearing in Equation (7.5.12) should fulfill the constraint $w_1 + w_2 = 1$.

Let the mse of the local estimation be represented as $E(\epsilon_L^2)$, and the mse of the frequency offset estimation done at the remote host be represented as $E(\epsilon_R^2)$. Then the mse of the combined frequency offset estimation $E(\epsilon_C^2)$ is shown to be

$$w_1^2 \times E(\epsilon_L^2) + w_2^2 \times E(\epsilon_R^2) \;=\; E(\epsilon_C^2) \;. \tag{7.5.13}$$

By expressing one of the coefficients as a function of the other, i.e. $w_2 = 1 - w_1$, and substituting the result in Equation (7.5.13), Equation (7.5.13) contains now only one unknown. To find the pair of weighting coefficients such that the resulting mse is minimum, the partial derivative is taken with respect to w_1, i.e. $\frac{\partial}{\partial w_1}\left\{w_1^2 \times E(\epsilon_L^2) + (1-w_1)^2 \times E(\epsilon_R^2)\right\} = \frac{\partial}{\partial w_1}\{E(\epsilon_{C_{min}}^2)\}$. The solutions for w_1 and w_2 are shown to be

$$w_1 = \frac{E(\epsilon_R^2)}{E(\epsilon_L^2) + E(\epsilon_R^2)} \;, \text{ and } w_2 = \frac{E(\epsilon_L^2)}{E(\epsilon_L^2) + E(\epsilon_R^2)} \;. \tag{7.5.14}$$

The behavior of a typical conversation pattern between two users is simulated. The top user in Figure 7.5.4, speaks most of the time until 3 quarters of the time interval considered, while the bottom user, listens at the beginning and starts talking at the end. It is obvious that the user who listens most of the time will have a better estimation of the frequency offset than the other user. However, by combining the frequency offset estimations done by both users, an improved frequency offset estimation is achieved to be utilized by both users. The coefficients that each of the individual frequency offset estimations need to be multiplied are given by Equation (7.5.14).

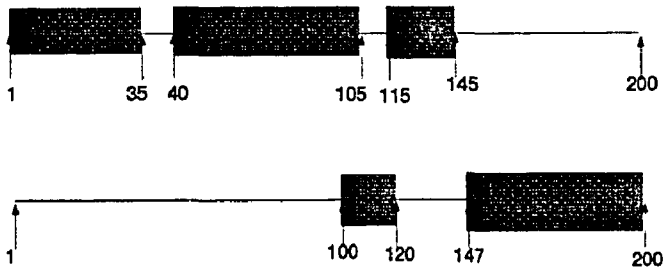

Figure 7.5.4 Packets received / sent between two communicating entities in a typical conversation pattern.

We assume that earlier the received packets for each individual user were properly divided into two subsets (as shown in Figure 7.5.5 (a), for the top user $\Gamma = 22$, while for the bottom user $\Gamma = 70$), and at this point our only concern is using simulations to validate Equation (7.5.14) regarding to the coupling of the two frequency offset estimations. Having the estimations ($E(\epsilon_L^2)$ and $E(\epsilon_R^2)$) of the two users, we vary w_1, ($w_2 = 1 - w_1$) from 0 to 1 (from 1 to 0) and note the values of w_1, w_2 that make the combined frequency offset estimation in Equation (7.5.12) minimum. In Figure 7.5.5 (b), the mean square error of the combined frequency offset estimation is plotted as a function of w_1. The minimum mean square error attainable using simulations occurs for $w_1 = 0.74$ which is in agreement with the value of $w_1 = 0.7392$ derived using Equation (7.5.14).

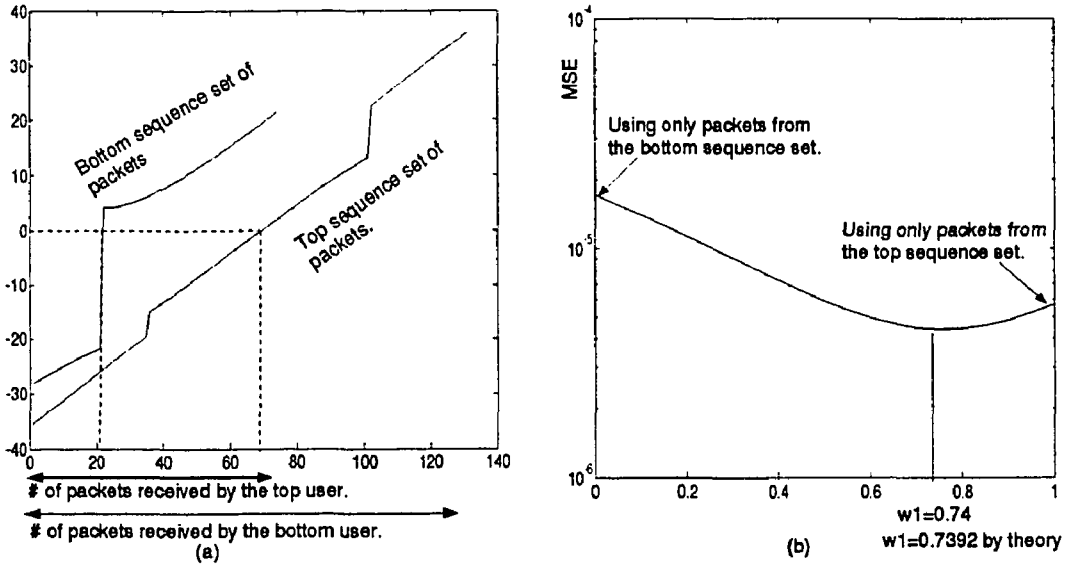

Figure 7.5.5 (a) Optimum division of each individual sequence of silence-talkspurt intervals in Figure 7.5.4 into two subsets according to Equation (7.6.15). (b) Geometrical representation of the shifting of the mean square error of the frequency offset estimation in response to the shifting of the coefficients $w_1, (w_2 = 1 - w_1)$ from 0 to 1 (1 to 0) .

In principle, the same approach taken for estimations done at different hosts can also be taken for estimations done locally at each host for two different media. However, for the case of video packets the above procedure should be applied with extreme care since the length of each video packet will vary in size due to coding. Then, the arrival time of the first fragment of the video packet should be taken as its own arrival time. If this cannot be achieved due to inability to access the IP layer (or this layer where the packet has not been re-assembled yet), then the results will be misleading since the delay of a packet is also a function of its size.

7.6 Optimum Division into Two Subsets of a Set of N Packets

In this section, the optimum way to divide a set of N packets into two sets that will lead to a minimum mean square error in the estimation of the frequency offset is provided. The division of the two sets will be in such a way that all packets arrived before

a determined time will belong to the same subset, and all packets arrived later than this time will belong to the other subset. These N packets were received in $N+k$ time slots, and whenever a mapping between the sequence number which represents the sequence the packet was transmitted by the sender and the sequence number which represents the time slot the packet was generated at the sender, the notation n_γ , $\gamma = 1, 2, ..N$, will be used. For instance, the sequence number of the time slot of the $\Gamma^{'th}$ packet transmitted by the sender will be n_Γ, and the sequence number of the time slot of the $N^{'th}$ transmitted packet will be n_N. In the specific case that this packet was generated at the last time slot $N+k$, then $n_N = N + k$.

To find the optimum way to divide the whole set $\mathfrak{N}$, the value of Γ which makes the partial derivative of the mean square error in Equation (7.4.1.10) with respect to Γ equal to zero, i.e. $\frac{\partial}{\partial \Gamma}(E(\epsilon^2)) = 0$, should be found. This is done in appendix G and there is shown that the following relation should be satisfied[3]

$$\min_{\Gamma} \left\{ \left| n_\Gamma - \left\{ \frac{1}{2(N-\Gamma)} \sum_{n \in \mathfrak{N}} n + \frac{1}{2} \left[\frac{1}{\Gamma} - \frac{1}{N-\Gamma} \right] \sum_{n \in \mathfrak{N}_\Gamma} n \right\} \right| \right\} \quad . \tag{7.6.15}$$

For the special case of continuous periodic arrival of packets Equation (7.6.15) leads to $\Gamma = N - \Gamma = \frac{N}{2}$ and $n_\Gamma = \frac{1}{N} \sum_{n=1}^{N} n = \frac{N+1}{2}$. Equation (7.6.15) has no closed form solution and has to be solved by trial and error. Nonetheless, this does not reduces its importance because the calculations involved are limited. As proven in appendix H, Equation (7.6.15) is an always increasing function in Γ, and therefore a unique solution exists. In addition, this implies that only $\lceil \log_2 N \rceil$ iterations are needed for its solution.

---

[3] Since n_Γ is an integer and the l.h.s of Eq. (G.5) is real, most probably the equality sign will not be held. For this reason, in Eq. (7.6.15) we take as Γ the value of Γ which is the closest to the value given by the l.h.s of Eq. (G.5).

7.6.1 Simulation Example.

Simulations that validate the theoretical results of this section are now presented. Assume that a source generated packets in a manner similar to the one shown in Figure 7.6.6. The shaded areas are talkspurt intervals whereas the rest are silent.

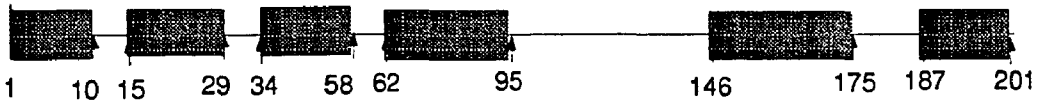

Figure 7.6.6 Silence-Talkspurt Intervals of the simulation example.

Solving Equation (7.6.15) by trial and error for the sequence of silence-talkspurt intervals shown in Figure 7.6.6, as shown in Figure 7.6.7 the value of Γ which leads to the minimum mean square error is 82. In order to validate Equation (7.6.15) we also resort to simulations whose results are displayed in Figure 7.6.8. There the mean square error of the frequency offset estimation is plotted as Γ varies from 11 to 121 , i.e. the first subset includes packets from 1 to Γ, and the second subset includes packets from $\Gamma + 1$ to N. Figure 7.6.8 shows two plots of the mean square error: one derived from the simulations, and one derived from Equation (7.4.1.10). Both plots indicate that the minimum mean square error is derived when the first subset includes packets up to $\Gamma = 81$, which validates Equation (7.6.15).

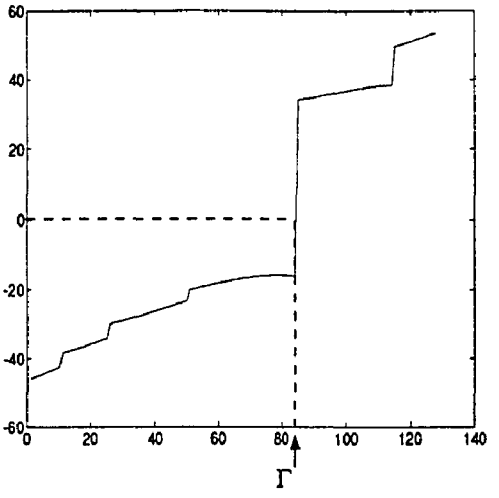

Figure 7.6.7 Solution of Equation (7.6.15) by trial and error.

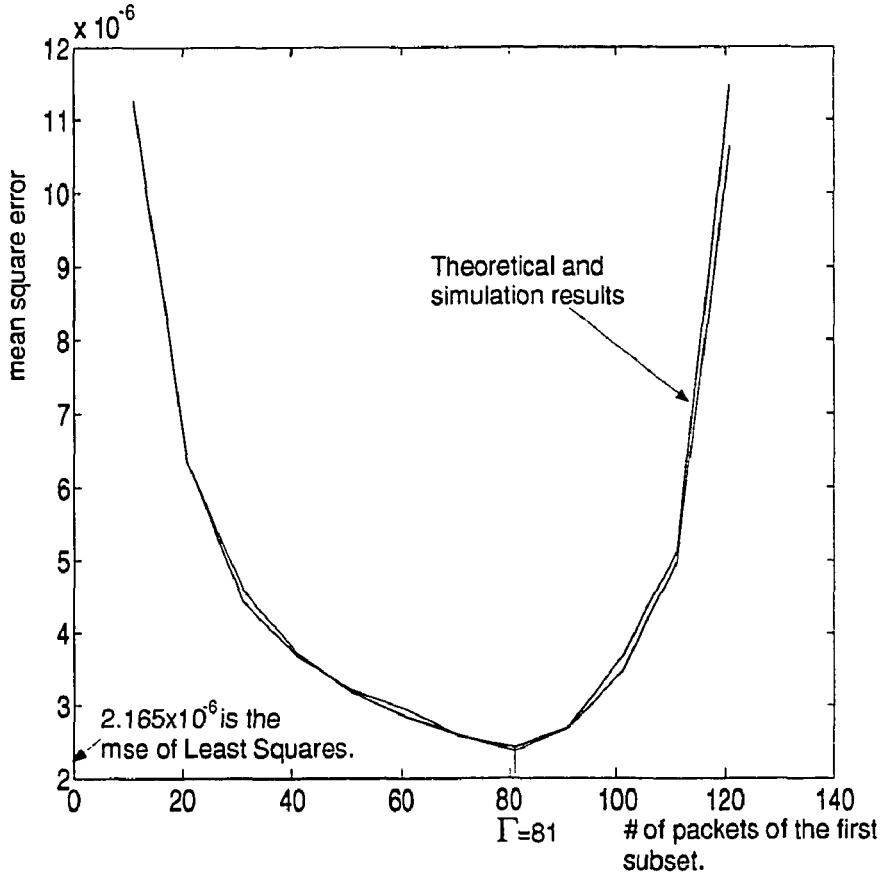

Figure 7.6.8 Theoretical and simulation results of the mean square error of the frequency offset estimation when the number of packets of the first subset Γ change from 11 to 121. The plot shown validates both Equations (7.6.15) $\min_{\Gamma}\left\{\left|n_{\Gamma}-\frac{1}{2(N-\Gamma)}\sum_{n\in\mathfrak{N}}n-\frac{1}{2}\left[\frac{1}{\Gamma}-\frac{1}{N-\Gamma}\right]\sum_{n\in\mathfrak{N}_{\Gamma}}n\right|\right\}$ and (7.4.1.10) $E(\epsilon^2)=\frac{\sigma_{\Delta}^2}{(\epsilon_2-\epsilon_1)^2(T^R)^2}\left(\frac{1}{\Gamma}+\frac{1}{N-\Gamma}\right)$.

7.6.2 Analysis of the Algorithm w.r. to the Computational Cost

Most of the computations are required for the solution of Equation (7.6.15). The number of computations required to compute the terms $\sum_{n\in\mathfrak{N}} n$ and $\sum_{n\in\mathfrak{N}_{\Gamma}} n$ are N additions plus N store instructions. These computations are executed independently of the number of the iterations. Then for each iteration the computations needed are 2 multiplications, 3 additions and one comparison. Therefore, the total number of computations is $Nadds + Nstores + \lceil \log_2 N \rceil \times (2mult + 3adds + 1comp)$. For the least squares approach most

of the computations are needed to construct the pseudo-inverse matrix $H^T H$, which are N additions and N multiplications. A fixed number of computations are also needed to take the inverse of the matrix $\left(H^T H\right)^{-1}$. Besides the cost involved in evaluating the frequency offset itself, there is also cost involved in determining the mean square error of the estimation. With our algorithm, the cost in evaluating the mse is very low (3 multiplications and 3 additions), but for the least squares algorithm N multiplications have to be performed (see Equation (7.4.1.11)).

7.7 Concluding Remarks for Chapter 6.

In this Chapter, an algorithm to estimate the frequency offset between two computers in real-time network applications was presented. The frequency offset is assumed to be a linear function of time. Accurate estimation of the frequency offset and its mean square error function enables the host participating in a real-time interaction to improve its synchronization with the remote host, thereby increasing the quality of service (QoS). The performance of the algorithm with respect to the mean square error as criterion is shown to be very close to the optimal derived from the least squares algorithm. Although for different sequence of packets the mean square error functions change, for a particular example our algorithm provides an estimation whose mse is 1% higher than the mse provided by the least squares algorithm. By combining the two estimations of the frequency offset done both at the local and at the remote host, firstly, a lower error is achieved, and secondly, a host that may have received only very few packets or none during the interval of consideration has also a good estimation of the frequency offset.

The computational cost of the proposed algorithm is much lower than the minimum mean square error (least squares) algorithm. Roughly, the proposed algorithm requires N additions compared to N multiplications required by the minimum mean square error

algorithm. The computational advantage of the algorithm is increased when evaluation of the mean square error of a particular estimation is also needed. In such a case, the proposed algorithm requires a handful of additions and multiplications compared to N multiplications required by the least squares algorithm.

Chapter 8 Conclusion

8.1 The Multimedia Conference Software

In this Thesis, the problem of Multimedia Synchronization in all of its aspects was addressed. We have concentrated ourselves more to the Multimedia Synchronization problems arising in applications such as Multimedia Conferencing, where all of the participants are active, in contrast to applications such as Video Store type of Applications, where only one of the participants is actually active and the server (the video store) is simply serving the users data (voice, video, or text) stored in its memory.

In the second Chapter, the Multimedia Conference software that we built in the Computer Telecommunications Laboratory at the City College of New York was presented. I would like to note here, that this was a big effort and many other Ph.D candidates took part in this work. The Multimedia Conference software that we built is capable of transmitting in "real-time" images, voice, text, and graphics data into a remote machine. Because the video card that we have used is video-pix and is capable only of four (4) frames per second, both images and the voice packets are transmitted every 250ms, which makes our multimedia conference not so real-time. This software was built on Sun workstations IPC running the operating system SUNOS 4.1.1 Rev. B. As a result, the graphical user interface was built using X-window programming, and the interprocess communication using socket programming.

We designed the software so that it utilizes two channels. One channel uses the TCP/IP protocol, and is used for transferring text and graphic information. The other channel uses the UDP/IP protocol, and through this channel the transfer of the real-time data is taking place, i.e video and voice packets.

After we built the multimedia conference software, we did some experiments to find the time delay and compare the TCP with the UDP protocol. It was found that the TCP/IP protocol is not suited for the transmission of real-time data due to the large time delay incurred to the packets during transmission. Part of the multimedia conference software appears in ICC '94 [2].

8.2 The Multimedia Synchronization Transport Protocol

In Chapters 4, 5 and 6, our proposed Multimedia Synchronization Transport Algorithm which is to be utilized for applications such as real-time multimedia conferencing was presented. Specifically, in Chapter 4 we presented the Point-to-Point Single-Medium Multimedia Synchronization Block, in Chapter 5 the Intermedia Synchronization Block, and in Chapter 6 the Inetrparticipant Synchronization Block.

In more detail, the issues discussed in Chapter 4 about the Point-to-Point Single-Medium Multimedia Synchronization Block were under the assumption that the traffic is noncontinuous periodic. The model of the traffic is noncontinuous periodic which nicely suits the voice traffic with silent-talkspurt intervals. Factors influencing synchronization are network jitter or variability of the delay and frequency offset between the clocks at the sender and at the receiver. The definition of the network jitter and its estimation is carried out using the concept of the reference times introduced first in [33, 32, 65]. In addition, a scheme that permits the receiver to uniquely identify the sequence number of each packet using a minimum number of bits and its proper implementation are presented. The optimal number of bits for a point-to-point communication considering only a single medium is found by exploiting the fact that the maximum time duration the numbering scheme should span needs not to be greater than the maximum variation of the jitter expected in the network. This work has been also submitted for publication in [53, 54]

In Chapter 5, two issues are explored in depth: optimal sequence numbering and minimum waiting time for playback. It is proven that the time interval the optimal sequence numbering scheme for Intermedia Synchronization should span needs not to cover a time duration greater than the maximum of the following two expressions: the maximum variation of the network jitter and twice the maximum absolute difference between the expected time delays of the two media. The concept of the reference times or expected arrival times that we have introduced in Chapter 4 is also used in this paper. As a consequence, many issues such as estimation of the reference times and bounds of the network jitter, are assumed to be known. Two different expressions for the minimum waiting time for the voice packets are derived: a minimum waiting time suitable when only the voice connection is active, and another when packets of both media are being transmitted. Finally, a switching mechanism that hunts for the minimum waiting time exhibiting the minimum delay suitable to the proper situation (voice transmission with or without video transmission) is presented. This work has been submitted for publication in [45, 46]

And in Chapter 6, synchronization issues arising in a multimedia multi-party conference were addressed. Specifically we proposed an algorithm to determine the set of packets generated periodically from different participants that are arriving at a node, either for mixing at the master of a conference, or for simply playing back at a regular participant of a conference. No global synchronization of the clocks was assumed. In Chapter 6 the statistical approach rather than the deterministic is used to determine the proper set of packets that have to be mixed at a given time. The statistics are derived from the sequence number which each packet carries along with it. The essence of the proposed algorithm is to estimate the average packet arrival time (or reference time) for each participant. With the reference time at hand, the maximum jitter and the optimum

waiting time for a mixer to wait packets from all participants can be determined. Both the accuracy and the computational complexity of the algorithm are improved by predicting the time / frequency offsets existing between the clocks at the source and the mixer. By employing feedback, the proposed algorithm is shown to lead to an optimum waiting / delay time. It is optimum in the sense that it is the same waiting / delay time that packets would experience in the case of only two participants, i.e. one sender and one receiver. The error of the proposed algorithm is enumerated by the Chernoff bound and demonstrated by simulation, and is shown to be acceptable in practical application. This work has also been published in [33] and submitted for publication also to [32, 34, 65]

8.2.1 Advantages / Disadvantages

There are many ***advantages*** of the proposed Multimedia Synchronization Transport Protocol. These are:

1. Modular Approach. Each of the three blocks functions independently of the others.
2. It is Comprehensive. It includes Point-to-Point Single Medium Communication, Intermedia Synchronization and Interparticipant Synchronization.
3. Exhibits the Optimum Delay. The delay is optimum because packets from each connection exhibit the maximum delay seen only in their own connection.
4. Deletion/Addition of each participant and/or media connection can happen independently (with no effect) to the other participants and or media connections.
5. No Global Network Clock Synchronization is assumed to exist.
6. Uses the minimum number of Control Bits for Sequence Numbering.
7. No Feedback Packets are required for Synchronization. Thus, No Complexity in the Implementation.

8. Computational Cost of the Algorithm is minimum. Only 2 to 3 additions per packet received!
9. It is Adaptive. It responds very fast (1 or 2 seconds) to changes in the network, i.e. load of the network, availability of computing resources in the gateways, etc.
10. It is Robust. The reference times or expected time delays are immune (or almost) to the network jitter.
11. The Protocol May Sit on the Application Layer. It is independent of the network infrastructure (FDDI, token ring, ethernet, TCP/IP, ATM, etc.). In short, Works Always and Everywhere.

There is one ***disadvantage*** however, and this has to do with the initiation time of the algorithm. It needs around 50 packets at the beginning to establish the Reference Times and the Bounds of the Network Jitter. Only after the reference times of the packets and the bounds of the network jitter have been established, the correct playback time for each packet can be found.

8.3 Estimation of the Frequency Offset in Real-Time Network Applications.

Finally, Chapter 7 is devoted exclusively to the issue of estimating the frequency offset in real-time network applications. The presented methodology provides the receiver with a mechanism to estimate the generation period of the packets at the sender $\widehat{T}^s$. knowledge of $\widehat{T}^s$ both lowers the frequency of the execution and improves the synchronization of the algorithm presented in [53, 54] and in Chapter 4. The frequency offset estimation is based on evaluating the tangent among two subsets of the arrived packets which are constructed so they lead to minimum mean square error. The estimation is further improved by combining the different estimations done locally at each host. The coupling of the estimations from different hosts provides adequate estimation

to the frequency offset also to the participant who has not received any packets, or very few, during the time interval in consideration. The computational cost of the proposed algorithm is significantly lower than the least squares approach, while its mean square error is only a fraction (in the simulation example only 1%) higher than the optimum. All theoretical results are validated using simulations, and the packet arrival times are assumed to be independent. The work presented in Chapter 7 has also been submitted for publication in [52, 30].

Appendix A Exact Representation of the Error in Estimating the Jitter of the x 'thPacket

In this appendix, the exact representation of the error due to the frequency offset the receiver makes in evaluating the jitter Δ_x of the x'^{th} packet is shown. The usefulness of this proof arises from the fact that no matter how far away the arrival time t_x of the packet x is from time X where the time offset contributions of the packets are zero, there is going to be no additional error in the evaluation of the reference times. We will proceed as follows: first we will show that if $t_x = X$, then the error due to the frequency offset in determining the jitter of the x'^{th} packet $\widehat{\Delta}_x$ is almost zero compared to Δ_x itself. Secondly, the exact error due to the frequency offset in the case $t_x \neq X$ will be shown. For ease of reference, Equation (4.4.8) is replicated below

$$\widehat{\Delta}_x = \frac{1}{N} \sum_{(n \in \mathfrak{N})} \left\{ t_x + (n-x)T^R - t_n \right\} - \mathcal{E} \,. \tag{A.1}$$

From Equations (4.3.2) and (4.3.3), the packet arrival times can be represented as follows

$$t_n = t_{ref,n} + \Delta_n = t_{ref,x} + (n-x)T^s + \Delta_n \,, \; n \in \mathfrak{N} \,, \tag{A.2}$$

where $\mathfrak{N}$ is the set of N time slots from 1 to $N+k$ that we have had packets transmitted from the sender. By substituting Equation (A.2) into Equation (A.1), we get the following

$$\begin{aligned} &\frac{1}{N} \sum_{(n \in \mathfrak{N})} \left[t_{ref,x} + \Delta_x + (n-x)T^R - \left\{ t_{ref,x} + \Delta_x + (n-x)T^s \right\} \right] = \\ &\qquad \frac{1}{N} \sum_{(n \in \mathfrak{N})} \left[\Delta_x - \Delta_n + (n-x)\left(T^R - T^s\right) \right] . \end{aligned} \tag{A.3}$$

Assuming for the moment that no frequency offset exists, the third term of Equation (A.3) is zero leading to

$$\frac{1}{N} \sum_{(n \in \mathfrak{N})} (\Delta_x - \Delta_n) = \Delta_x - \frac{1}{N} \sum_{(n \in \mathfrak{N})} \Delta_n \overset{N \to \infty}{\to} \Delta_x \,. \tag{A.4}$$

Here, we have to be careful to make the distinction that the range of the set $\mathfrak{N}$ can grow to infinity, i.e. $N + k \rightarrow \infty$, but the number of actual packet arrivals N might be very small or zero. This may very well be a realistic case if one of the participants is not speaking at all.

To take into account the frequency offset between the sender and the receiver, and remembering that we assumed for the moment that t_x and X coincide, the third term can be rewritten as

$$\frac{1}{N}\sum_{(n\in\mathfrak{N}_E)}(n-x)\Big(T^R-T^s\Big)+\frac{1}{N}\sum_{(n\in\mathfrak{N}_L)}(n-x)\Big(T^R-T^s\Big)=\mathcal{E}\Rightarrow$$

$$-\frac{1}{N}\sum_{(n\in\mathfrak{N}_E)}(x-n)T^s+\frac{1}{N}\sum_{(n\in\mathfrak{N}_L)}(n-x)T^s=\mathcal{E}\frac{T^s}{(T^R-T^s)}\Rightarrow$$

$$-\frac{1}{N}\sum_{(n\in\mathfrak{N}_E)}(t_x-t_n)+\frac{1}{N}\sum_{(n\in\mathfrak{N}_L)}(t_n-t_x)+\frac{1}{N}\sum_{(n\in\mathfrak{N})}\Delta_n+\frac{N_L-N_E}{N}\Delta_x=\mathcal{E}\frac{T^s}{(T^R-T^s)}\ . \tag{A.5}$$

Since we assumed that $t_x = X$, applying Equation (4.4.6) to the third line of Equation (A.5), the first two terms cancel out. Therefore, when $t_x = X$ the error in estimating Δ_x due to frequency offset is negligible and can be considered zero

$$\mathcal{E}=\frac{(T^R-T^s)}{T^RN}\left[-\sum_{n\in\mathfrak{N}}\Delta_n+(N_L-N_E)\Delta_x\right]\sim 0\ . \tag{A.6}$$

Having proved the above, we now proceed in determining exactly the error we are making when $t_x \neq X$.

Let us take the time distance $X - t_x$, which might be positive or negative depending on whether the closest packet arrival time t_x is earlier or later than time X. We are guaranteed by way of construction that in a time distance $|X - t_x|$ around point X no packet arrival occurred. Also, as a reminder, the point X is the point where the contributions of the time offsets of all the received packets cancel out. This means that at time t_x the *extra* time offset contributions of the packets which belong to the set $\mathfrak{N}_E$,

i.e $n \in \mathfrak{N}_{\mathrm{E}} \Rightarrow \forall n\ t_n \leq X$, will equal to

$$\sum_{(n \in \mathfrak{N}_{\mathrm{E}})} (X - t_x) \frac{(T^R - T^s)}{T^R} \,. \tag{A.7}$$

Similarly, for the received packets belonging to the set $\mathfrak{N}_{\mathrm{L}}$, their extra time offset contributions will equal to

$$\sum_{(n \in \mathfrak{N}_{\mathrm{L}})} (X - t_x) \left(\frac{T^R - T^s}{T^R} \right) . \tag{A.8}$$

Therefore, the accumulated extra *average* time offset contributions of all the received packets to time t_x will equal to

$$\frac{1}{N} \left(\sum_{n \in (\mathfrak{N}_{\mathrm{E}} \cup \mathfrak{N}_{\mathrm{L}})} \right) (X - t_x) \left(\frac{T^R - T^s}{T^R} \right) = \ (X - t_x) \left(\frac{T^R - T^s}{T^R} \right) \ . \tag{9}$$

The average contribution of the time offsets is evaluated in Equation (9) above because the average quantity is the one which appears in Equation (A.1) in the determination of the network jitter. Also in the summation sign, someone might be tempted to subtract the point x from the set $\mathfrak{N} = \mathfrak{N}_{\mathrm{E}} + \mathfrak{N}_{\mathrm{L}}$ because its contribution to its arrival time t_x is zero. However, this will be invalid because we are really interesting in the time offset contribution that the packet x has subtracted in the time interval $|X - t_x|$. X again is the point where the time offset contributions of *all* the packets are zero.

There is an obvious similarity between Equations (A.7), (A.8), (9) and Equation (A.5). In Equation (A.5) the two terms from the *l.h.s.* can be viewed as the total time offset contributions of the two sets $\mathfrak{N}_{\mathrm{E}}$ and $\mathfrak{N}_{\mathrm{L}}$ to time t_x, while Equations (A.7), (A.8), (9) represent only the extra time offsets contributions of the two sets . In other words, the time offset contributions to t_x have been broken to contributions to time X, which cancel out by construction of point X, and to contributions in the time interval $X - t_x$.

Appendix B Proof of the Theorem Related to Finding the Optimal Sequence Numbering Scheme for Intermedia Synchronization.

We will show the validity of the theorem by showing that the condition displayed in Equation (5.2.1.3) fulfills the requirements stated. For convinience, Equation (5.2.1.3) is repeated below

$$\Psi = \max\left\{2\left|D^{vi} - D^{vo}\right|_g\ ,\ (\Delta_{max})_g - (\Delta_{min})_g\right\} \tag{B.1}$$

Requirement 1) Since Ψ is the maximum of the two expressions in Equation (B.1), this implies that $\Psi \geq (\Delta_{max})_g - (\Delta_{min})_g \Rightarrow 2^{p_\psi} \times T^{vo} \geq \Psi \geq (\Delta^{vo}_{max})_g - (\Delta^{vo}_{min})_g$. Thus, for both media steams, voice and video, the receiver is able to recognize the sequence number of the packets with only p_ψ bits. Let us for example take the case of the voice packets. We will show that the time interval Ψ is large enough so that the receiver will be able to recognize the sequence number of any packet. This requirement in effect implies that the receiver should be able to distinguish packets that were generated at different times but they carry the same p_ψ-bit sequence number. We will denote by t^{vo}_n the arrival time of the packet that its sequence number is needed to be recognized at the receiver, and as t^{vo}_m , $m \in \mathfrak{M}$ the arrival times of the voice packets that have identical p_ψ-bit sequence numbers, i.e. $(t^{vo}_n)_{p_\psi} = (t^{vo}_m)_{p_\psi}$ but $m \neq n$. This in effect means that the elements of the set $\mathfrak{M}$ are generated as follows $m = n \pm k2^{p_\psi}$, $k = 1, 2, \ldots$. It suffices to distinguish the $n^{'th}$ packet from any other packet with identical p_ψ-bit sequence number if for every time the difference of their generation times $g^{vo}_n - g^{vo}_m > 0$, then $t^{vo}_n - t^{vo}_m > 0$, or every time $g^{vo}_n - g^{vo}_m < 0$, then $t^{vo}_n - t^{vo}_m < 0$.

It is clear that if the conditions presented above are satisfied for one immediate neighbor of n, i.e. $m = n \pm 2^{p_\psi}$, then it will hold for all other $m \in \mathfrak{M}$. Let us have $m = n + 2^{p_\psi}$. Then, $g_n^{vo} - g_m^{vo} = g_n^{vo} - \left(g_n^{vo} + 2^{p_\psi} T^{vo}\right) = -2^{p_\psi} T^{vo} < 0$. Therefore, we have to prove that $t_n^{vo} - t_m^{vo} < 0$ whenever $2^{p_\psi} T^{vo} \geq \left(\Delta_{max}^{vo}\right)_g - \left(\Delta_{min}^{vo}\right)_g$. We have:

$$t_n^{vo} - t_m^{vo} = t_{ref,n}^{vo} - t_{ref,m}^{vo} + \Delta_n^{vo} - \Delta_m^{vo} = t_{ref,n}^{vo} - \left(t_{ref,n}^{vo} + 2^{p_\psi} T^{vo}\right) + \Delta_n^{vo} - \Delta_m^{vo} < 0 \Rightarrow$$
$$2^{p_\psi} T^{vo} > \max\left\{\Delta_n^{vo} - \Delta_m^{vo}\right\} = \max\left\{\Delta_n^{vo}\right\} - \min\left\{\Delta_m^{vo}\right\} = \left(\Delta_{max}^{vo}\right)_g - \left(\Delta_{min}^{vo}\right)_g \quad \text{(B.2)}$$

In Equation (B.2) we have used $t_{ref,m}^{vo} = t_{ref,n}^{vo} + 2^{p_\psi} T^{vo}$ since $g_n^{vo} - g_m^{vo} = -2^{p_\psi} T^{vo}$ and $t_{ref,n}^{vo} = g_n^{vo} + D^{vo}$, $t_{ref,m}^{vo} = g_m^{vo} + D^{vo}$.

Requirement 2) According to the mechanism presented in Figure 5.2.3, although the generation periods of the two media streams are different, i.e. $T^{vi} = zT^{vo}$, $z = 1, 2, ...,$ the sender attaches the same p_ψ-bit sequence number to both packets (one packet from each medium) in case their generation times are the same. We would like the receiver to be able to pair any video packet to the corresponding voice packet whose generation time is the same with that of the video packet. This in effect implies that the receiver should be able to identify that voice packet whose p_ψ-bit sequence number matches that of the video packet and its generation time is the same as that of the video from the set $\mathfrak{M}$ of voice packets that carry the same p_ψ-bit sequence number with that of the video packet, i.e. $m \in \mathfrak{M} \Rightarrow \forall m \in \mathfrak{M}, \quad \left(t_m^{vo}\right)_{p_\psi} = \left(t_n^{vi}\right)_{p_\psi}$.

We would like to note here that before the search of the pair of packets —one from each medium — that were generated at the same time at the sender starts, the receiver has already established the reference times for each medium. What is unknown is the difference in the expected time delays between the two media $D^{vi} - D^{vo}$. Since

the reference times for both media have been established, it suffices for the receiver to guarantee that

$$\left|t_{ref,n}^{vi} - t_{ref,n}^{vo}\right| = \min_{m\in\mathfrak{M}} \left\{\left|t_{ref,n}^{vi} - t_{ref,m}^{vo}\right|\right\} \tag{B.3}$$

It is intuitively clear that if the receiver is able to distinguish the $n^{'th}$ voice packet (the voice packet that was generated at the same time with the $n^{'th}$ video packet) from its immediate neighbors $m^- = n - 2^{p_\psi}$ and $m^+ = n + 2^{p_\psi}$, then it will be able to distinguish the $n^{'th}$ voice packet from any other voice packet which belongs to the set $\mathfrak{M}$. Let us represent the generation times of the video packet and the three voice packets as $g_n^{vi}, g_n^{vo}, g_{m^-}^{vo}, g_{m^+}^{vo}$, where the following relations hold $g_n^{vo} = g_n^{vi}$, $g_{m^+}^{vo} = g_n^{vo} + 2^{p_\psi}T^{vo}$, $g_{m^-}^{vo} = g_n^{vo} - 2^{p_\psi}T^{vo}$. Let us also represent the reference times of these packets as $t_{ref,n}^{vi}$, $t_{ref,n}^{vo}$, t_{ref,m^-}^{vo} , t_{ref,m^+}^{vo}. Then the following relations between the generation times of these packets and their corresponding reference times hold $t_{ref,n}^{vi} = g_n^{vi} + D^{vi}$, $t_{ref,n}^{vo} = g_n^{vo} + D^{vo}$, $t_{ref,m^-}^{vo} = g_{m^-}^{vo} + D^{vo}$, $t_{ref,m^+}^{vo} = g_{m^+}^{vo} + D^{vo}$.

Without loss of generality, let us have $D^{vi} > D^{vo}$. Then, for the pair of packets that were generated at the same time, $t_{ref,n}^{vi} - t_{ref,n}^{vo} = D^{vi} - D^{vo}$, and for the immediate neighbors (m^-) and (m^+) we have $t_{ref,n}^{vi} - t_{ref,m^-}^{vo} = g_n^{vi} + D^{vi} - \left[g_n^{vo} - 2^{p_\psi}T^{vo} + D^{vo}\right] = D^{vi} - D^{vo} + 2^{p_\psi}T^{vo}$ and $t_{ref,n}^{vi} - t_{ref,m^+}^{vo} = D^{vi} - D^{vo} - 2^{p_\psi}T^{vo}$. Thus, since $D^{vi} > D^{vo}$, $D^{vi} - D^{vo} + 2^{p_\psi}T^{vo} > D^{vi} - D^{vo} - 2^{p_\psi}T^{vo}$, meaning that it is sufficient to prove that $\left|D^{vi} - D^{vo}\right| < \left|D^{vi} - D^{vo} - 2^{p_\psi}T^{vo}\right|$. However, $D^{vi} > D^{vo}$ and assuming that $2^{p_\psi}T^{vo} > \left|D^{vi} - D^{vo}\right|$, we arrive at

$$D^{vi} - D^{vo} < 2^{p_\psi}T^{vo} - \left(D^{vi} - D^{vo}\right) \Rightarrow 2^{p_\psi}T^{vo} > 2\left(D^{vi} - D^{vo}\right) \tag{B.4}$$

Similarly, in case that $D^{vi} < D^{vo}$, $\left|D^{vi} - D^{vo} - 2^{p_\psi} T^{vo}\right| > \left|D^{vi} - D^{vo} + 2^{p_\psi} T^{vo}\right|$, meaning that it is sufficient to prove that $\left|D^{vi} - D^{vo}\right| < \left|D^{vi} - D^{vo} - 2^{p_\psi} T^{vo}\right|$. However, assuming that $2^{p_\psi} T^{vo} > \left|D^{vi} - D^{vo}\right|$, we arrive at

$$\begin{aligned} \left|D^{vi} - D^{vo}\right| < \left|D^{vi} - D^{vo} + 2^{p_\psi} T^{vo}\right| \Rightarrow \; & -\left(D^{vi} - D^{vo}\right) < D^{vi} - D^{vo} + 2^{p_\psi} T^{vo} \\ \Rightarrow \; & 2^{p_\psi} T^{vo} > \; -2\left(D^{vi} - D^{vo}\right) \end{aligned} \tag{B.5}$$

Combining Equations (B.4) and (B.5), we have the following expression for the time interval Ψ the sequence numbering scheme should span

$$2^{p_\psi} T^{vo} > \Psi > 2\left|D^{vi} - D^{vo}\right| \overset{\max}{\rightarrow} 2\left|D^{vi} - D^{vo}\right|_g \tag{B.6}$$

□

Appendix C Reference Time Estimation When The Traffic Is Continuous Periodic.

In this appendix we show the validity of Equation

$$\widehat{\Delta}^i_{\lceil \frac{N}{2} \rceil} = \frac{1}{N} \sum_{n=1}^{N} \left(t^i_{\lceil \frac{N}{2} \rceil} + \left(n - \left\lceil \frac{N}{2} \right\rceil \right) T^R - t^i_n \right) . \tag{C.1}$$

Then we show why we have used the packet arrival time of the $\lceil \frac{N}{2} \rceil$'th packet, $t^i_{\lceil \frac{N}{2} \rceil}$, and not any other packet arrival time.

We assume for a moment that there are no frequency offsets between the clocks at the sources and the clock at the receiver. We will prove that in the limit the *r.h.s* of Equation (C.1) approaches $\Delta^i_{\lceil \frac{N}{2} \rceil}$, which exactly represents the amount of time the $\lceil \frac{N}{2} \rceil \ th$ packet arrival time $t^i_{\lceil \frac{N}{2} \rceil}$ is off from its reference time $t^i_{ref,\lceil \frac{N}{2} \rceil}$. From Equations

$$t^i_{ref,n} = t^i_{ref,1} + (n-1)T^i \tag{C.2}$$

and

$$\Delta^i_n = t^i_n - t^i_{ref,n} \tag{C.3}$$

, the packet arrival times t^i_n can be represented as function of $t^i_{ref,\lceil \frac{N}{2} \rceil}$ and $\Delta^i{}_n$ as follows:

$$t^i_n = t^i_{ref,n} + \Delta^i_n = t^i_{ref,\lceil \frac{N}{2} \rceil} + \left(n - \left\lceil \frac{N}{2} \right\rceil \right) T^i + \Delta^i_n \ , \ 1 \le n \le N. \tag{C.4}$$

As it was also noted in Equation (6.2.1), even though source *i* sends packets with nominal period T, for the observer at the receiver it appears that the sender transmits packets with period T^i. By substituting Equation (C.4) into Equation (C.1), the *r.h.s* of Equation (C.1) becomes

$$\frac{1}{N} \sum_{n=1}^{N} \left[t^i_{ref,\lceil \frac{N}{2} \rceil} + \Delta^i_{\lceil \frac{N}{2} \rceil} + \left(n - \left\lceil \frac{N}{2} \right\rceil \right) T^R - \left\{ t^i_{ref,\lceil \frac{N}{2} \rceil} + \Delta^i_n + \left(n - \left\lceil \frac{N}{2} \right\rceil \right) T^i \right\} \right] =$$

$$\frac{1}{N} \sum_{n=1}^{N} \left[\Delta^i_{\lceil \frac{N}{2} \rceil} - \Delta^i_n + \left(n - \left\lceil \frac{N}{2} \right\rceil \right) \left(T^R - T^i \right) \right] \tag{C.5}$$

Since frequency offsets are not considered yet, T^R=T^i and the third term in Equation (C.5) is zero leading to

$$\frac{1}{N}\sum_{n=1}^{N}\left(\Delta^i_{\lceil\frac{N}{2}\rceil}-\Delta^i_n\right) = \Delta^i_{\lceil\frac{N}{2}\rceil}-\frac{1}{N}\sum_{n=1}^{N}\Delta^i_n \overset{N\to\infty}{\to} \Delta^i_{\lceil\frac{N}{2}\rceil} \tag{C.6}$$

since the average value of Δ^i_n approaches zero by definition of the network jitter. Thus Equation (C.1) has been proved. Notice that the error of our estimation is the term $\epsilon = \frac{1}{N}\sum_{n=1}^{N}\Delta^i_n$ due to finite N considered.

The careful reader may already start wondering why the packet arrival time $t^i_{\lceil\frac{N}{2}\rceil}$ was chosen and not any other packet arrival time, for instance the first t_1 or the last t_n. We choose the middle point of the time interval [0,NT], or the middle packet $\lceil\frac{N}{2}\rceil$ because only at this point the contributions of the time offsets (caused by the frequency offset of the corresponding clock at the source and the clock at the receiver) in the estimation of $\Delta^i_{\lceil\frac{N}{2}\rceil}$ cancel themselves out. The proof is as follows: If T^R *is not* T^i Equation (C.5) will result in one more term, the following

$$\frac{1}{N}\sum_{n=1}^{N}\left(n-\left\lceil\frac{N}{2}\right\rceil\right)\left(T^R-T^i\right) =$$

$$= \left(T^R-T^i\right)\left(\frac{N+1}{2}-\left\lceil\frac{N}{2}\right\rceil\right) = \begin{cases}\dfrac{(T^R-T^i)}{2} & N\ even\\ 0 & N\ odd\ .\end{cases} \tag{C.7}$$

The effect of the third term is approximately zero as shown above. For comparison purposes, if we use the first packet arrival time t^i_1 instead of $t^i_{\lceil\frac{N}{2}\rceil}$ in Equation (C.1), the third term can be shown to be

$$\frac{1}{N}\sum_{n=1}^{N}(n-1)\left(T^R-T^i\right) = \left(T^R-T^i\right)\frac{(N-1)}{2}, \tag{C.8}$$

and Equation (C.1) now becomes

$$\widehat{\Delta}^i_1+\left(\frac{N-1}{2}\right)\left(T^R-T^i\right) = \frac{1}{N}\sum_{n=1}^{N}\left(t^i_1+(n-1)T^R-t^i_n\right)\ . \tag{C.9}$$

For example, if N=101 and $(T^R - T^i) = 10^{-3} \times T^R$, the time offset is $0.05 \times T^R$. As exemplified with Equation (C.7), we better take N odd integer to avoid any frequency offset contribution.

Appendix D Confidence Interval

In this appendix we give a justification to the argument that when the number of packets used in the estimation of the average is increased by a factor of k, then the confidence interval is reduced by $\sqrt{k}$. Using the *Chebyshev's inequality*, we can show that this is true when the bound used is the bound derived by it, i.e.

$$P\left\{\left|\frac{1}{Nk}\sum_{n=1}^{Nk}\Delta_n^i\right| \geq \frac{\epsilon}{\sqrt{k}}\right\} \leq \frac{\frac{\sigma_\Delta^2}{Nk}}{\left(\frac{\epsilon}{\sqrt{k}}\right)^2} = \frac{\sigma_\Delta^2}{N\epsilon^2} \text{ and } P\left\{\left|\frac{1}{N}\sum_{n=1}^{N}\Delta_n^i\right| \geq \epsilon\right\} \leq \frac{\sigma_\Delta^2}{N\epsilon^2} \tag{D.1}$$

where the Δ_n's are zero mean and assumed to be *i.i.d.* random variables with variance $\sigma^2{}_\Delta$. Δ_n's. Chebyshev's inequality is a loose upper bound and the reader may question the validity of the argument when the Chernoff bound is used (In Chapter 5, the Chernoff bound is used exclusively for determining an upper bound for the probability of error). For this reason, some remarks are in order. Firstly, the validity of the argument is proven readily — however, in an ad-hoc method — for the Chernoff bound for the case of the uniform density function by trying a few examples. Secondly, the validity of the argument, not only that the confidence interval is reduced $\sqrt{k}$ but also the probability of error for the two cases ($P\left(\frac{\epsilon}{\sqrt{k}}\right)$ *and* $P(\epsilon)$) are exactly the *same*, is proved for the normal probability density function using the Central Limit Theorem (see p. 108 in [62]). In addition, using the Central Limit Theorem, which gives an upper bound for the probability of error lower than the bound derived from Chebyshev's inequality but slightly bigger than the Chernoff bound, the argument we are making is true for *any* probability density function when the bound considered is the bound derived from the Central Limit Theorem.

Appendix E Cancellation of the Time Offsets

In this appendix we show that the average value obtained by averaging kN packet arrival times which are affected by the presence of a linear-with-time frequency offset is true only for the middle time of the interval into consideration. . For example, the time instance kNT of the time interval [0,kNT] cannot be taken as the time that the average value applies. The reason is that the contribution of the time offsets of each of the k group averages over N packets would be non-zero and actually equal to

$$\sum_{n=1}^{k} T_{off}^{i}|_{t=nNT} = \sum_{n=1}^{k} \left(f^{i} - f^{R}\right) nNT = \left(f^{i} - f^{R}\right) NT \sum_{n=1}^{k} n \qquad \text{(E.1)}$$

where f^R is assumed to be different than f^i. But if we take the middle time kNT/2 , the contribution is zero as it is clearly seen below

$$\sum_{n=-\frac{k}{2}}^{\frac{k}{2}} T_{off}^{i}|_{t=nNT} = \left(f^{i} - f^{R}\right) NT \sum_{n=-\frac{k}{2}}^{\frac{k}{2}} n = 0 \qquad \text{(E.2)}$$

Appendix F Mean Square Error of the Frequency Offset Estimation

The estimation of the frequency offsets is done every 2kN packets received. For simplicity of the notation, we assume that k=1. Therefore, 2N packets are grouped into 2 groups, one group with the first N packets and the second with the last N packets. Using Equation $\Delta_n^i = t_n^i - t_{ref,n}$ and the identity $T^i = T^R - (T^R - T^i)$, we can define the normalized packet arrival time τ_n^i as follows

$$\tau_n^i = t_n^i - t_{ref,\lceil \frac{N}{2} \rceil}^i - \left(n - \left\lceil \frac{N}{2} \right\rceil\right) T^R = -\left(n - \left\lceil \frac{N}{2} \right\rceil\right)\left(T^R - T^i\right) + \Delta_n^i . \quad \text{(F.1)}$$

The *p.d.f.* of the random variables x and y, where $x = \frac{1}{N}\sum_{n=1}^{N} \tau_n^i$ and $y = \frac{1}{N}\sum_{n=N+1}^{2N} \tau_n^i$ will be very close to Normal according to the Central Limit Theorem, assuming that the Δ_n^i's are independent and N is sufficiently large. This can be easily seen if for example, x is written as follows:

$$x = \frac{1}{N}\left[-\left(T^R - T^i\right)\sum_{n=1}^{N}\left(n - \left\lceil \frac{N}{2} \right\rceil\right) + \sum_{n=1}^{N}\Delta_n^i\right] \Rightarrow$$

$$x = \begin{cases} \dfrac{-(T^R - T^i)}{2} + \dfrac{1}{N}\displaystyle\sum_{n=1}^{N}\Delta_n^i & , \; N \text{ is even} \\ \dfrac{1}{N}\displaystyle\sum_{n=1}^{N}\Delta_n^i & , \; N \text{ is odd} \; . \end{cases} \quad \text{(F.2)}$$

Since the term $-(T^R - T^i)/2$ is very small, it can be neglected. Thus, the Central Limit Theorem can now be applied to the normalized sum of N *i.i.d.* Δ_n^i's. The variance of the new random variables x and y will be $\sigma_x^2 = \sigma_y^2 = \sigma_\Delta^2/N$, where σ_Δ^2 is the variance of the Δ_n^i's . Given the uncertainty caused by the Δ_n^i's , we find the mean square error in estimating the frequency offset $(T^R - T^i)$.

The error in estimating the frequency offset arises from the fact that although the two groups of points have the same probability density function, their exact distance from their true average will be different. We emphasize here that where both groups give the same distance from their mean , i.e. $x-\bar{x}=y-\bar{y}$, there will be no error in the frequency offset estimation. Since the error in the estimation does not depend on the individual distances that the random variables x and y are away from their true mean but rather on the relative distance between them, the error will be

$$\epsilon=\frac{|x-y|}{N}\,. \tag{F.3}$$

It can be shown easily that for any two independent Normal Random variables x and y with variance $\sigma_x^2=\sigma_y^2=\sigma_\Delta^2/N$, and mean zero, the *p.d.f.* of the new random variable x-y is a zero mean Normal random variable with variance

$$\sigma_{x-y}^2=2\sigma_x^2=\frac{2\sigma_\Delta^2}{N}\,. \tag{F.4}$$

Therefore, the mean square error in determining the frequency offset using our methodology is

$$E(\epsilon^2)=E\left[\left(\frac{|x-y|}{N}\right)^2\right]=\frac{E\left[(x-y)^2\right]}{N^2}=\frac{2\sigma_\Delta^2}{N^3}\,, \tag{F.5}$$

and making the transformation $N\rightarrow\frac{N}{2}$ for the case of only N packets received, we derive to the final expression for the mean square error

$$E(\epsilon^2)=\frac{16\sigma_\Delta^2}{N^3}\,. \tag{F.6}$$

Appendix G Partial Derivative of the Mean Square Error Function.

The expression of the mean square error in estimating the frequency offset is given by Equation (7.4.1.10). The term of the denominator $(\xi_2 - \xi_1)^2$ in Equation (7.4.1.10) can be rewritten as follows

$$\left[\frac{1}{N-\Gamma}\sum_{n\in\mathfrak{N}_{N-\Gamma}} n - \frac{1}{\Gamma}\sum_{n\in\mathfrak{N}_{\Gamma}} n\right]^2 = \left[\frac{1}{(N-\Gamma)\Gamma}\left\{\Gamma\sum_{n\in\mathfrak{N}} n - N\sum_{n\in\mathfrak{N}_{\Gamma}} n\right\}\right]^2 \qquad \text{(G.1)}$$

Substituting Equation (G.1) in Equation (7.4.1.10) we get

$$E(\epsilon^2) = \frac{\sigma_{\Delta}^2 N(N-\Gamma)\Gamma}{\left[\Gamma\sum_{n\in\mathfrak{N}} n - N\sum_{n\in\mathfrak{N}_{\Gamma}} n\right]^2 (T^R)^2} \quad . \qquad \text{(G.2)}$$

In order to take the derivative of the term $\sum_{n\in\mathfrak{N}_{\Gamma}} n$, *i.e.* $\frac{\partial}{\partial\Gamma}\left(\sum_{n\in\mathfrak{N}_{\Gamma}} n\right)$ we use the result in p. 349 in [66]. There is shown that the derivative D operator of a discrete value function can be approximated by

$$D = \frac{1}{h}\ln(1+\Delta) = \frac{1}{h}\left(\Delta - \frac{\Delta^2}{2} + \frac{\Delta^3}{3} - \ldots\right) \simeq \frac{\Delta}{h} \quad , \qquad \text{(G.3)}$$

where Δ is the value of the difference of the function in two consecutive points, i.e. $\Delta = f(\Gamma+1) - f(\Gamma)$, and h is the difference of the running variable, i.e. $h = (\Gamma+1) - \Gamma$. By simply applying the definition of the derivative, we have

$$\frac{\partial}{\partial\Gamma}\left(\sum_{n\in\mathfrak{N}_{\Gamma}} n\right) = \frac{\partial}{\partial\Gamma}[f(\Gamma)] = \frac{f(\Gamma+1) - f(\Gamma)}{1} = n_{\Gamma} \quad . \qquad \text{(G.4)}$$

Having showed the above, we proceed by taking the partial derivative of Equation (G.2) *w.r.* to Γ in order to find the value of Γ leading to minimum mean square estimation of the frequency offset. We have

$$\frac{\partial}{\partial\Gamma}\{E(\epsilon^2)\} = 0 \Rightarrow (N-2\Gamma)\left[\Gamma\sum_{n\in\mathfrak{N}} n - N\sum_{n\in\mathfrak{N}_\Gamma} n\right] = 2\Gamma(N-\Gamma)$$
$$\left[\sum_{n\in\mathfrak{N}} n - Nn_\Gamma\right] \Rightarrow n_\Gamma = \frac{1}{2(N-\Gamma)}\sum_{n\in\mathfrak{N}} n + \frac{1}{2}\left[\frac{1}{\Gamma} - \frac{1}{N-\Gamma}\right]\sum_{n\in\mathfrak{N}_\Gamma} n \quad . \tag{G.5}$$

Appendix H Proof that the Solution of Equation (7.6.15) Requires $\lceil \log_2 N \rceil$ Iterations

If we prove that the function inside the absolute sign in Equation (7.6.15) is a always increasing function in the variable Γ, then it follows that we have a unique solution of Equation (7.6.15). In addition, only $\lceil \log_2 N \rceil$ iterations are needed to find the solution. The proof is as follows. For any Γ, the following is true

$$n_\Gamma - \left\{ \frac{1}{2(N-\Gamma)} \sum_{n \in \mathfrak{N}} n + \frac{1}{2}\left[\frac{1}{\Gamma} - \frac{1}{N-\Gamma}\right] \sum_{n \in \mathfrak{N}_\Gamma} n \right\} = C_\Gamma \quad , \tag{H.1}$$

where C_Γ is any constant that will make the equation valid. Then for the $\Gamma + 1$ point, we have

$$n_{\Gamma+1} - \left\{ \frac{1}{2(N-\Gamma-1)} \sum_{n \in \mathfrak{N}} n + \frac{1}{2}\left[\frac{1}{\Gamma+1} - \frac{1}{N-\Gamma-1}\right] \sum_{n \in \mathfrak{N}_{\Gamma+1}} n \right\} = C_{\Gamma+1} \quad , \tag{H.2}$$

If we prove that $C_{\Gamma+1} - C_\Gamma > 0 \quad \forall \Gamma, \ \Gamma = 1, 2, .., N$, then this ends the proof of the statement. Let us subtract the *l.h.s* of Equation (H.1) from the *l.h.s* of Equation (H.2). We get

$$\begin{aligned}
&n_{\Gamma+1} - n_\Gamma + \frac{1}{2}\left[\frac{1}{N-\Gamma} - \frac{1}{N-\Gamma-1}\right] \sum_{n \in \mathfrak{N}} n + \frac{1}{2}\left[\frac{1}{\Gamma} - \frac{1}{\Gamma+1} + \frac{1}{N-\Gamma-1} - \frac{1}{N-\Gamma}\right] \\
&\sum_{n \in \mathfrak{N}_\Gamma} n + \frac{1}{2}\left[\frac{N-2\Gamma-2}{(\Gamma+1)(N-\Gamma-1)}\right] n_{\Gamma+1} = \quad n_{\Gamma+1} - n_\Gamma + \frac{1}{2}\left\{\frac{1}{(N-\Gamma)(N-\Gamma-1)}\right\} \\
&\left\{ \sum_{n \in \mathfrak{N}_\Gamma} n + \sum_{n \in \mathfrak{N}_{N-\Gamma}} n \right\} + \frac{1}{2}\left\{\frac{1}{\Gamma(\Gamma+1)} + \frac{1}{(N-\Gamma)(N-\Gamma-1)}\right\} \sum_{n \in \mathfrak{N}_\Gamma} n + \\
&\frac{1}{2}\left[\frac{N-2\Gamma-2}{(\Gamma+1)(N-\Gamma-1)}\right] n_{\Gamma+1} \geq \frac{1}{2}\left\{\frac{2}{(N-\Gamma)(N-\Gamma-1)} + \frac{1}{\Gamma(\Gamma+1)}\right\} \sum_{n \in \mathfrak{N}_\Gamma} n + \\
&n_{\Gamma+1}\left\{1 + \frac{1}{2}\left[\frac{N-2\Gamma-2}{(\Gamma+1)(N-\Gamma-1)}\right] + \frac{1}{2}\left[\frac{1}{(N-\Gamma-1)}\right]\right\} - n_\Gamma = \\
&\frac{1}{2}\left\{\frac{2}{(N-\Gamma)(N-\Gamma-1)} + \frac{1}{\Gamma(\Gamma+1)}\right\} \sum_{n \in \mathfrak{N}_\Gamma} n + n_{\Gamma+1}\left\{1 + \frac{1}{2(\Gamma+1)}\right\} - n_\Gamma > 0
\end{aligned} \tag{H.3}$$

since $n_{\Gamma+1} - n_\Gamma > 0 \ \forall \Gamma, \ \Gamma = 1, 2, .., N$.

REFERENCES

[1] Tarek N. Saadawi, M. Ammar, and A. Elhakeem . *"Fundamentals of Telecommunications Networks "*. John Wiley & Sons, New York, 1994.

[2] Weiqiang Bai, Panagiotis N. Zarros, Myung J. Lee and Tarek Saadawi. "Design and Analysis of a Multimedia Conference System Using TCP/UDP". Proceedings of *ICC '94*,pp. 432-438.

[3] Brian Field and Taieb Znati. "Experimental Evaluation of Transport Layer Protocols For Real Time Applications". *Proceeding of 16th Local Computer Networks Conference*,pp.521-534, 1991.

[4] W. Doeringer, D. Dykeman, M. Kaiserswerth, B. W. Meister, H. Rudin and R. Williamson . "A survey of Light Weight Transport Protocols for High Speed Networks". *IEEE Transactions on Commun., Vol. 38, No 11*, ,pp. 2025-2039, Nov. 1990.

[5] H. Shimizu, M. Mera and H. Tani . "Packet Communication Protocol for Image Services on a High-Speed Multimedia LAN ". *IEEE Journal on Selected Area in Commun.,Vol. 7, No 5* ,pp.782-788, June 1989.

[6] A. Netravali, W. D. Roome, and K. Sabnani . "Design and Implementation of a High Speed Transport Protocol". *IEEE Transactions on Commun.,Vol. 38,No 11* ,pp. 2010-2023, Nov 1990.

[7] Myung Lee, Yue He and Tarek Saadawi. "Multimedia Integration on FDDI". *Proceedings of IEEE Network Management and Control Workshop 1993*,pp. 234-239.

[8] Ralf Steinmetz. "Synchronization Properties in Multimedia Systems". *IEEE Journal on Selected Area in Commun.*, Vol. 8,pp. 401-412, April 1990.

[9] Shiro Sakata. "Development and Evaluation of an In-House Multimedia Desktop Conference System ". *IEEE J. on Selected Areas in Commun.*,Vol. 8,No 3,pp. 340-346, April 1990.

[10]Wu-Hon Leung, Thomas Baumgartner, Yeou Hwang, Mike Morgan, and Shi-Chuan Tu. "A Software Architecture for Workstations Supporting Multimedia Conferencing in Packet Switching Networks ". *IEEE J. on Selected Areas in Commun.*,Vol. 8,No 3,pp. 380-389, April 1990.

[11]Robert E. Blumberg and David H. Walters. "A PC-Based Multimedia Document Manager ". *IEEE J. on Selected Areas in Commun.*,Vol. 7,No 2, pp. 283-288, Feb. 1989.

[12]P.V. Rangan, H. M. Vin, and S. Ramanathan. "Communication Architectures and Algorithms for Media Mixing in Multimedia Conferences". *IEEE/ACM Transactions on Networking*,Vol.1, pp. 20-47, Feb. 1993.

[13]P. V. Rangan,H. M. Vin, and S. Ramanathan. "Designing an on Demand Multimedia Service". *IEEE Communications magazine*,Vol.30,pp. 56-65, July 1992.

[14]S. Ramanathan and P.V. Rangan. "Adaptive Feedback Techniques for Synchronized Multimedia Retrieval over Integrated Networks". *IEEE/ACM Transactions on Networking*, Vol.1,pp.246-260, April 1993.

[15]G. Barberis, and D. Pazzaglia . "Analysis and Optimal Design of a Packet Voice Receiver ". *IEEE Trans. in Communications, COM-39*,,pp. 217-227, Feb. 1980.

[16]W. A. Montgomery . "Techniques for Packet Voice Synchronization ". *IEEE Journal on Selected Areas in Communications*,No 6, pp. 1022-1027, Dec. 1983.

[17]F. Alvarez-Cuevas, M. Bertran, F. Oller, and J. M. Selga. "Voice Synchronization in Packet Switching Networks ". *IEEE Network magazine,Vol. 7 No 5* ,pp. 20-25,

Sep. 1993.

[18]R. Ramjee, Jim Kurose, Don Townsley, and Henning Schulzrinne . "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks ". Proceedings of *IEEE Infocom 94* ,pp. 680-688.

[19]Ibrahim Habib and Tarek N. Saadawi. "Multimedia Traffic Characteristics in Broadband Networks". *IEEE Communications magazine*, Vol. 30,pp 70-77, July 1992.

[20]Debasis Mitra . "Network Synchronization: Analysis of a Hybrid of Master-Slave and Mutual Synchronization ". *IEEE Transactions on Communications,Vol.Com-28,No 8* ,pp. 1245-1259, August 1980.

[21]D. Mills. "Internet Time Synchronization: The Network Time Protocol". *IEEE Transactions on Communications*,Vol.39,pp. 1482-1493, Oct. 1991.

[22]William C. Lindsey, and Anil V. Kantak . "Network Synchronization of Random Signals ". *IEEE Transactions on Communications,Vol. Com-28,No 8* ,pp. 1260-1266, August 1980.

[23]Algimantas Kajackas . "On Synchronization of Communication Networks with Varying Channel Delays ". *IEEE Transactions on Communications,Vol. Com-28,No 8* ,pp. 1267-68, August 1980.

[24]John C. Luetchford, Jan Heymen, and J. W. Bowick . "Synchronization of a Digital Network ". *IEEE Transactions on Communications,Vol. Com-28,No 8* ,pp. 1285-1290, August 1980.

[25]Panagiotis N. Zarros, Myung J. Lee, and Tarek N. Saadawi. "Synchronization Algorithms In Transporting Real-Time Multimedia Data Over Computer Networks: A Tutorial". to be submitted to *ACM/Springer-Verlag Multimedia Systems Journal* .

[26]Martin de Prycker. *Asynchronous Transfer Mode solution for broadband ISDN*. 1992 by Ellis Horwood Limited, Chichester, England.

[27]Rene L. Cruz . "A Calculus for Network Delay, Part I: Network Elements in Isolation ". *IEEE Transactions on Information Theory, Vol 37, No 1* ,pp. 114-131, Jan. 1991.

[28]Wassim Matragi, Chatschik Bisdikian, and Koshrow Sohraby . "Jitter Calculus in ATM Networks: Single Node Case ". Proceedings of *IEEE Infocom '94* ,pp. 232-241.

[29]Zheng Wang and Jon Crowcroft . "Analysis of Burstiness and Jitter in Multimedia Communications ". Proceedings of *IEEE GLOBECOMM '93, Austin, Texas* ,pp. 1496-1500.

[30]Panagiotis N. Zarros, Myung J. Lee, and Tarek N. Saadawi. "Frequency Offset Estimation In a Packet Switching Environment". submitted to *IEEE Transactions on Communications* .

[31]Panagiotis Nikolaos Zarros . *"Multimedia Network Synchronization In real-Time Applications "*. Ph.D. Thesis, Graduate Center, The City University of New York, Nov. 1994.

[32]Panagiotis N. Zarros, Myung J. Lee, and Tarek N. Saadawi . "Interparticipant Synchronization in Real-Time Multimedia Conference Using Feedback ". to appear in the *IEEE/ACM Transactions on Networking* .

[33]Panagiotis N. Zarros, Myung J. Lee, and Tarek N. Saadawi . "Statistical Synchronization Among participants in Real-Time Multimedia Conference ". Proceedings of *IEEE Infocom '94, Toronto CA*,pp. 912-919.

[34]Panagiotis N. Zarros, Myung J. Lee, and Tarek N. Saadawi. "A Synchronization Algorithm for Distributed Multimedia Environments". to be published in *ACM/Springer-Verlag Multimedia Systems Journal.*

[35]Panagiotis N. Zarros, Myung J. Lee, and Tarek N. Saadawi . "Statistical Synchronization Among Participants in Real-Time Multimedia Conferencing ". to appear in *Journal of Networks and Systems Management* .

[36]T. Little and A. Ghafoor. "Multimedia Synchronization Protocols for Broadband Integrated Services ". *IEEE J. Selected Areas Commun.*, Vol. 9,pp. 1368-1382, Dec. 1991.

[37]T. Little and A. Ghafoor. "Synchronization and Storage Models for Multimedia Objects ". *IEEE J. Selected Areas of Commun.*,Vol. 8,pp. 413-427, April 1990.

[38]Miae Woo, Naveed U. Qazi, and Arif Ghafoor . "A Synchronization Framework for Communication of Pre-orchestrated Multimedia Information ". *IEEE Network,vol 8, No. 1,pp. 52-61*, Jan. 1994.

[39]Miae Woo and Arif Ghafoor . "Multichannel Scheduling for Communication of Pre-orchestrated Multimedia Information ". Proceedings of *IEEE Infocom 94, Toronto Ca.* ,pp. 920-928.

[40]L. Li, A. Karmouch, and N.D. Georganas . "Multimedia Segment Delivery Scheme and Its Performance for Real-Time Synchronization Control ". Proceedings of *IEEE Multimedia '94*,pp. 1734-1738.

[41]K. Ravindran and V. Bansal. "Delay Compensation Protocols for Synchronization of Multimedia Data Streams". *IEEE Transactions on Knowledge & Data Engineering*,vol.5,pp. 574-589, Aug. 1993.

[42]J. Escobar, C. Patridge, and D. Deutsch. "Flow Synchronization Protocol". *IEEE/ACM Transaction on Networking*,vol. 2,pp. 111-121, April 1994.

[43]Taieb Znati and Brian Field. "A Network Level Channel Abstraction For Multimedia Communication in Real-Time Networks". *IEEE Transactions on Knowledge & Data*

Engineering,vol. 5,pp. 590-599, Aug. 1993.

[44]R. Stevens . *"UNIX Networking Programming ". Prentice Hall, NJ* , 1990.

[45]Panagiotis N. Zarros, Myung J. Lee, and Tarek N. Saadawi. "Synchronization in Real-Time Network Applications, Part II: Intermedia". submitted to *IEEE Journal on Selected Areas in Communications.*

[46]Panagiotis N. Zarros, Myung J. Lee, and Tarek N. Saadawi. "A Multimedia Synchronization Transport Algorithm in Real-Time Applications, Part II: Intermedia ". submitted to *IEEE Infocom '95.*

[47]D. Bertsekas and R. Gallager . *"Data Networks* ". Englewood Cliffs, NJ Prentice Hall, 2nd Edition, 1992.

[48]Rene L. Cruz . "A Calculus for Network Delay, Part II: Network Analysis ". *IEEE Transactions on Information Theory, Vol 37, No 1* ,pp. 132-141, Jan. 1991.

[49]Israel Cidon, Asad Khamisy, and Moshe Sidi . "Dispersed messages in Discrete-Time Queues: Delay, Jitter and Threshold Crossing ". Proceedings of *IEEE Infocom '94* ,pp. 218-223.

[50]Wassim Matragi, Chatschik Bisdikian, and Koshrow Sohraby . "Jitter Calculus in ATM Networks: Multiple Node Case ". Proceedings of *IEEE Infocom '94* ,pp. 242-252.

[51]H. Schulzrinne . "Sample Profile and Encodings for the Use of RTP for Audio and Video Conferences with Minimal Control ". *Internet-Draft* Expires 12/31/93.

[52]Panagiotis N. Zarros, Myung J. Lee, and Tarek N. Saadawi . "Efficient Estimation of the Frequency Offset in Real-Time Network Applications". submitted to *IEEE ICC '95.*

[53]Panagiotis N. Zarros, Myung J. Lee, and Tarek N. Saadawi. "Synchronization in Real-Time Network Applications, Part I: Point-to-Point ". submitted to *IEEE Journal on Selected Areas in Communications.*

[54]Panagiotis N. Zarros, Myung J. Lee, and Tarek N. Saadawi. "A Multimedia Synchronization Transport Algorithm in Real-Time Applications, Part I: Point-to-Point ". submitted to *IEEE Infocom '95.*

[55]Thomas D. C. Little, Arif Ghafoor . "Spatio-Temporal Composition of Distributed Multimedia Objects for Value-Added Networks ". *IEEE Computers, Special issue on Multimedia Info. Sys.* ,vol. 24,pp. 42-50, Oct. 1991.

[56]L. Li, A. Karmouch, and N.D. Georganas . "Real-Time Synchronization Control in Multimedia Distributed Systems ". Proceedings of *IEEE Multimedia '92.*

[57]Cormac J. Sreeman . "A Service Oriented Approach to Continuous Media Synchronization ". Proceedings of *IEEE Infocom '94* ,pp. 936-943.

[58]Chun-Chuan Yang and Jau-Hsiung Huang . "A real-time Synchronization Model and Transport Protocol for Multimedia Applications ". Proceedings of *IEEE Infocom '94* ,pp. 928-935.

[59]C. Papadopoulos and G.M. Parulkar. "Experimental Evaluation of SUNOS IPC and TCP/IP Protocol Implementation". *IEEE/ACM Transactions on Networking*, Vol. 1,pp. 199-216, April 1993.

[60]Jack K. Wolf and Jay W. Schwartz . "Comparison of Estimators for Frequency Offset ". *IEEE Transactions on Communications, Vol. 38,No 1* , pp. 124-127, Jan. 1990.

[61]Harold W. Sorenson . *"Parameter Estimation "*. Marcel Dekker, Inc.,New York, 1980.

[62]John Wozencraft and Irwin Mark Jacobs. *Principles of Communications Engineering.* 1965 by John Wiley & Sons , reissued in 1990 by Waveland Press.

[63]I. S. Gradshteyn and I. M. Ryzhik . *"Table of Integrals, Series, and Products "*. Academic Press,London, 1980.

[64]Louis L. Scharf . *"Statistical Signal Processing, Detection, Estimation and Time-Series Analysis "*. Addison Wesley, Massachusetts, 1991.

[65]Panagiotis N. Zarros, Myung J. Lee, and Tarek N. Saadawi . "Statistical Synchronization Among Participants In Real-Time Multimedia Conferencing ". to appear in the *Journal of Networks and Systems Management* .

[66]Mathews and Walker . *"Mathematical Methods of Physics "*. Addison Wesley Massachusetts, 2nd edition.