Koc University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master' s thesis by

Tahir Celebi

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Prof. Reha Civanlar (Advisor)

Assist. Prof. Oznur Ozkasap (Advisor)

Prof. Murat Tekalp

Assist. Prof. Engin Erzin

Assist. Prof. Serdar Tasiran

Date: ___________________________________

To my family,

## ABSTRACT

I present a distributed implementation of a novel approach to multipoint videoconferencing rooted in a peer-to-peer model of media transmission. In this thesis, principals of the peer-to-peer system that I developed are described together with its performance evaluation. Besides, a formal verification for the proposed approach has been done. The system focuses on the needs of participants with symmetric, low bandwidth connections to the Internet. It does not require additional hardware, as in Multipoint Control Units, or network infrastructure support such as multicast. With almost no additional demands on the networking and computing resources needed for a point-to-point videoconference, the new approach would be able to extend a point-to-point conference into a multipoint videoconference. In comparison to offered approach, I also include a brief survey on existing peer-to-peer solutions for interactive media applications.

Mostly, it is not fair to share equal responsibility among all users, since it is a well known fact that Internet provides an environment where various users (categorized according to the computing power or bandwidth capacity) interact together. So, for the sake of performance and solution applicability, it will be good to share responsibility among users according to their available resources. Thus, this thesis also offers an optimization over the proposed peer-to-peer video dissemination solution where routing is done considering the hosts' resources.

# ÖZET

Uçtan-uca (orj. *Peer-to-peer*) medya iletimini temel alan Çok-noktalı Video Konferans (orj. *Multipoint Videoconferencing*) uygulaması için olağan yaklaşımların dışında dağıtık bir uygulama sunuyorum. Bu tezde geliştirmiş olduğum Uçtan-uca sistemin prensipleri performans değerlendirmeleriyle beraber ele alınıyor. Ayrıca önerilen yaklaşım icin matematiksel bir ispatta yapılmaktadır. Sistem simetrik ve dar-band İnternet kullanıcıları üzerine odaklanmıştır. Sistem ne Çok-noktalı Kontrol Unite’ lerinde (orj. *Multi Control Unit*) olduğu gibi ek bir donanıma gereksinim duyar ne de ağ seviyesinde Çoka-gönderim (orj. *Multicast*) desteği ister. Yeni yaklaşım kullanıcılara ek bir destek gerektirmeksizin ( ağ desteği, güçlü makina desteği) Noktadan-noktaya Konferansı (orj. *Point-to-point Conference*) Cok-noktalı Konferanslara genişletebilir. Önerilen çözümle beraber etkileşimli medya uygulamalarının Uctan-uca çözümleriyle ilgili kısa bir inceleme de bu calışmada yer almaktadır.

Çok iyi bilindiği gibi İnternet birbirinden çok farklı kullanıcılar icin bir ortam sunmaktadır ki bahsedilen kullanıcılar yerel-hesaplama gücü, band kapasitesi gibi kriterlere göre kategorize edilebilirler. Genelde kullanıcılar arasında eşit sorumluluk paylaşımı bir önceki cümlede bahsedilen nedenden dolayı pek de adil değildir. Önerilen çözümün performasını ve uygulanabilirliğini arttırmak için sorumluluk kullanıcılar arasında dağıtılırken kapasitelerinin göz önüne alınması gerekir. Bu yüksek lisans tezi, önerilen çözüm üzerinde bazı iyileştirmeri katılımcıların kaynaklarını göz önünde bulundurarak yapmaktadır.

# ACKNOWLEDGEMENTS

I extend my sincere gratitude and appreciation to many people who made this master thesis possible. Special thanks are due to my supervisors Professor Reha Civanlar and Assist. Professor Oznur Ozkasap. I appreciate the confidence that they showed in my abilities and (especially) the patience they exhibited in waiting for me to get things done.

I am highly indebted to the colleagues from AtosOrigin Company just for accepting me to carry on my research at the hard-working hours. They gave me different visions for handling problems ranging from minor to major.

Many thanks go to one of my dearest friend M. Bahadir Soydan from the Mechanical Engineering Department of Koc University. He shared my problems and gave me strength with his valuable advices.

Special thanks go to my family since they showed me the value of hard work and always encouraged me to set my goals high. Especially, my father Professor Ilyas Celebi who believes me more than anyone else never drawbacks his spiritual and financial support ignoring his own problems. If this thesis satisfies its goals, the main reason is his eternal trust and encouragements.

Last but not the least, I would like to thank to the most Gracious and Almighty creator of everything. He blesses me with everything I own.

# TABLE OF CONTENTS

# LIST OF FIGURES

# NOMENCLATURE

| | |
|---|---|
| GA | Genetic Algorithm |
| GUI | Graphical User Interface |
| IM | Instant Messaging |
| ISDN | Integrated Services Digital Network |
| MCU | Multi Control Unit |
| MBone | Multicast Back Bone |
| MP | Multi Point |
| MPEG | Motion Pictures Experts Group |
| MST | Multicast Spanning Tree |
| P2P | Peer to Peer |
| RP | Rendezvous Point |
| RTP | Real Time Protocol |
| RTCP | Real Time Control Protocol |
| SPF | Single Point of Failure |
| ST | Spanning Tree |
| TCP | Transmission Control Protocol |
| TFRC | TCP Friendly Rate Control |
| TLA | Temporary Logic of Actions |
| TLC | Temporary Logic Checker |
| UDP | User Datagram Protocol |

*“The world is changing. I can smell it in the air; I can feel it in the water.”*
*Galadriel*

# Foreword

Heraclites (BC 536- 475), one of the most famous Greek philosophers, founder of the “panta rhei” (everything flows)[1] said that: “No one can step twice into the same river”. We can see this reality revealed centuries ago. Many paradigms exhibited in the several branches of the science have been replaced with the newer ones and this evolution will never stop until the end of the living-world. For instance; Newton’s mechanics has been replaced with the Quantum mechanics, secret of DNA has been explained, our perception for the universe has been changed by the discovery of black holes etc. With the help of thousand-yeared experience, we can smoothly say that each improvement triggers another one and this cycle will go on forever. Since the only stable thing in the universe is the “nothingness” itself, and mankind is also a part of this universe, he has also been affected from the evolution process. He cannot be thought apart from it. Not only the physical view (not talking about“evolution theory”)but also the mental perspective of the man has been altered in time. Therefore as the decades pass, requirements of the humans have been changed with the newer inventions and discoveries. Even this process is so fast that yesterday’s requirements have been changed today and they are replaced with the newer ones needed more complex solutions. Besides, number of the requirements increases, as the time passes. For instance; one decade ago, globalization might not be thought as a profound and an achievable topic for the whole world but today with the help of the improvements in the telecommunication’s world, it is seen as mandatory and has been nearly accomplished. But this does not mean that everything has been succeeded for the sake of globalization. Strongly possible, tomorrow will bring with different needs and today’s solutions for the globalization will be depreciated. But an interesting point here is that since human-being is the only smart

[1] See http://www.thebigview.com/greeks/heraclitus.html.

living thing in the universe, actually he is the one who manages the evolution steps. So as a consequence, everything in the evolution chain has a direct relation with the human being. This final result can be summarized as; mankind determines the directions of the evolution chain, with his needs, wants and demands.

Internet invented in the second period of the 20th century has showed many improvements until it gets its today' s form. However, with the great enhancement of the telecommunication world especially in physical lines for the last decade, quality of data transmission has been improved in a dramatic manner. New technologies like ADSL and Cable which are suitable for fast and reliable communication have been served for the end users. As an unavoidable result of the unstoppable evolution process, today Internet requires totally a new examination including its existence and content. Due to the fact that nowadays computers play an important role in all kinds of people' s life, it is an obligation for the Internet to query and re-built itself.

"The Internet is changing: Static content is giving way to streaming video, text is being replaced by music and the spoken word, and interactive audio and video is becoming commonplace. These changes require new solutions and applications and they pose new unique challenges for application designers." [2]

Tahir CELEBI

(Istanbul, 2004)

[2] Collin Perkins, "RTP: Audio and Video For the Internet", Boston, June 2003.

# 1 INTRODUCTION

## 1.1 Motivation and Goal

Mostly due to the large bandwidth demand of video transport and the associated high costs, the use of video conferencing in general and multipoint (MP) video conferencing in particular is not as popular as audio conferencing. As an example, today most of the IM (Instant messaging) vendors (for instance; MSN Messenger 6.x by Microsoft ® Corporation) equip their applications with pair-wise video conferencing functionality. In these applications, each added party increases both the outgoing and the incoming bandwidth requirements for all the end points involved. Given that low bandwidth connections e.g., modem connections over the phone lines, which still constitute a significant percentage of global access to the Internet, are barely enough for video communication to a single point, a MP video conferencing system implementation whose bandwidth demand increases as the participant count increases is not realistic.

As the requirements for video communication increase today, new and brilliant solutions have to be produced. Apart from the social needs, there are a number of factors that define the success of video conferencing in legacy and converged networks. In reality, these factors are valid for the other types of interactive applications. One approach to reduce the bandwidth requirement of a MP video conferencing system is using additional network based devices called multipoint control units (MCU) [1]. An MCU is a central device which has more bandwidth than a regular participant and receives all participants' video signals and disseminates them. An MCU prepares a video representation e.g., collages of low resolution video and sends it to all participants. Today for IP conferencing, MCUs are capable of some additional complex features like

IP precedence, IP type of services (TOS), intelligent packet loss recovery (IPLR), resource reservation protocol (RSVP) and total conference and call control. However, MCUs higher costs and complexity make them suitable to mostly larger business applications. Besides, using MCUs makes the end points (conference participants) thicker since MCUs generally requires H.323 compliant participants. Thus, it is not an easy job to deploy such a solution to everywhere since their high costs (e.g. complexity, price etc.) make them unsuitable for not only video conferencing but also other types of interactive applications.

Today the most well-known solution for many interactive applications including video conferencing is using Multicast Backbone (MBone) like tools at the transmission side when it is supported by the underlying network. An additional advantage of a multicasting-based solution is the reduced operation complexity due to its distributed nature [2]. Unfortunately, due to the fact that wide-scale deployment of native network mode multicasting on the global Internet has still not realized, this option is not a viable choice for many people either. Another deficiency for the multicast is the traditional approach of the network administrators. Closing multicast addresses is one of the conventional methods whenever traffic congestion occurs in the network. Besides, multicasting burdens the network devices like routers, so router' s operational responsibility increases. This causes middle-devices to be run with possible deficiencies.

This thesis targets to propose and present an alternative approach to MP video conferencing based on the use of a distributed peer-to-peer (P2P) system [3, 4]. The system has no need for additional hardware, as in MCUs, or network infrastructure support such as multicast. With almost no additional demand on the networking and computing resources needed for a point-to-point video conference, this system can make it possible to extend a point-to-point conference into a MP video conference, where each participant can see one other selected participant under most practical cases. The system also includes a simple conference management protocol used for the operations like user login, logout, etc. A packet video standard MPEG-4 which is generally used for high-resolution and interactive video has been outsourced from FFMPEG project which is freely licensed and can be downloaded from http://www.sourcefourge.com. A prototype of the proposed architecture was built by extending an H.263 based, point-to-point video telephony implementation to MP. The resulting system can be used for MP video conferencing among participants with modem connections to the Internet without requiring multicast support or MCUs and it can be used for extending most other point-to-point systems also. On the other hand, this thesis does not consider the issues like network address translation, security and integrating the solution with other types of networks except IP.

P2P techniques are becoming extremely popular and finding diverse applications. Use of P2P techniques for video conferencing through an end system based multicast implementation has been reported before [5]. However, such systems assume availability of larger upstream bandwidths for each participant. In [6], another P2P solution for MP video conferencing based on a centralized architecture is presented. Details on these approaches are available in the next chapter. Different from prior work, this study, by focusing on the needs of those users with symmetric, low bandwidth connections, presents a totally distributed solution with no single point of failure and central server maintenance.

How beautiful Marian Wright Edelman said: “If you don’ t like the way the world is, you change it. You just do it one step at a time”.

## *1.2 Contributions of Thesis*

Contributions of this thesis can be titled as follows;

- A fully distributed architecture for multipoint video conferencing where all peers are identical and have a restriction of “one-to-one I/O” constraint. This means that a peer just sends and receives at most 2 video packets (one for input channel and one for output channel) to minimize the bandwidth usage.

- To support such kind of mechanism, a novel approach “Chain Based Solution” has been offered. This thesis suggests a solution for finding feasible orderings for one-to-one constrained nodes while providing a MP Conferencing system.
- A product, Nebula Video Conferencing (NeViC) which is a complete application based on the above requirements has been designed, developed and deployed with its own conference signaling, configuration, transmission protocols and modules.
- The Internet is a heterogeneous environment. Different peers may have different computing power and network access, meaning different users should have different requirements. For instance; a user with limited computing power (CPU, bandwidth) may handle fewer responsibilities where the one with high computing power may take higher loads. Therefore, this thesis also takes this fact into consideration and proposes an extension for the system, namely looses the constraint as “one-to-many”. This means, peers still can receive at most one video packet at any given time, but output constraint changes depending on the network bandwidth (more bandwidth forces more responsibilities). Furthermore, the improvement gained from applying the optimization techniques to the whole system is measured via simulations.
- After developing the whole system as a complete application, it was verified mathematically utilizing a formal verification tool (TLC).
- This thesis does not offer a solution for convergent networks. Proposed solution is applied to and deployed at IP networks. It does not consider the issues like integration to different networks and making the system available for other types of networks like SS7, X.25 etc.

## *1.3 Thesis Outline*

The history of the audio/video conferencing used is overviewed in Chapter 2. Besides, some terminologies used in this thesis are described in Chapter 2. A literature survey of the video conferencing algorithms appropriate for use in MP video conferencing has been given in Chapter 3. In Chapter 4, detailed descriptions of the solution and architecture of the application are given. Chapter 5 describes the offered optimization techniques for the proposed solution. Finally, the thesis is concluded with results and future directions.

# 2 BACKGROUND

## 2.1 Introduction

Starting from the 80' s ofthe previous century, businesses started to utilize from teleconferencing using room systems over proprietary networks. After porting the H.32X standard into the Integrated Services Digital Network (ISDN), end-users had a new choice for video-conferencing. All of these enhancements leaded to profound investments, and required high management and operation costs. Next paragraph describes the unprecedented growth of video conferencing briefly;

"The dramatic growths of the Internet and Internet Protocol (IP) networks, together with the universal adoption of standards for multimedia conferencing over packet networks, have caused businesses to switch to not only voice conferencing but also videoconferencing over IP networks. The reduced cost of videoconferencing equipment, affordable desktop stations, cheaper bandwidth, ever-improving video quality and the availability of enhanced video-related services are driving businesses to adopt IP based conferencing as a cost-effective means to inter and intra organizational communications." [3]

On the other hand, implementing IP based conferencing solutions is less complex than implementing ISDN based solutions. At last but not the least, IP based approach serves a vision for communicating different networks transparently; turning legacy networks into convergent networks. Convergent networks ensure the communication of different applications at different networks without considering the network architecture. IP

[3] see http://www.broadcastpapers.com/broadbiz/RadvisionIPCentricConf01.htm

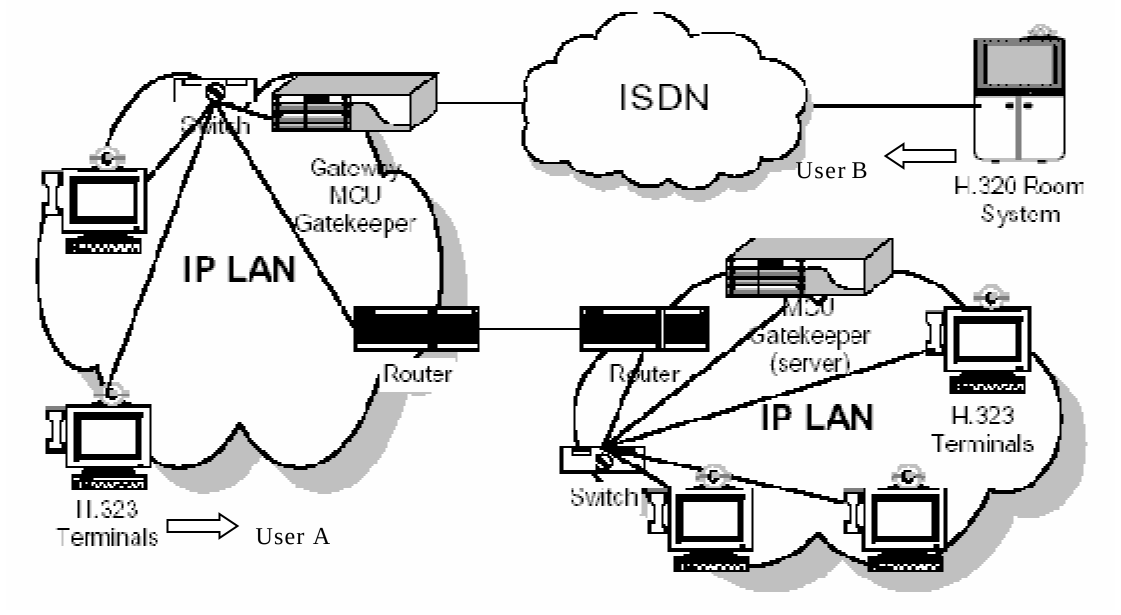


**Figure 2. 1** A converged approach for the IP conferencings

provides us a way to easily deploy and scale current solutions. Examine the converged conferencing application given in figure 2.1[4]. Although this figure includes too many details, there is no need to worry about it. In this scenario, end users are unaware of the end points. For instance, user-A registers for the conference from an intranet whereas user-B registers for the conference from public switched telephone network (PSTN). By looking at this graphic, it is seen that it is not an easy job to select a feasible solution for hybrid networks. Since each client residing at different networks have different performance requirements and service criteria, and each solution comes with its positive and negative affects. This chapter aims to clarify some of the issues and technologies associated with IP based conferencing.

## 2.2 IP Based Conferencing

IP based conferencing is real-time multimedia voice, video and data conferencing over IP networks. This section introduces some basic IP based conferencing terms and concepts.

[4] see http://www.broadcastpapers.com/broadbiz/RadvisionIPCentricConf02.htm

### 2.2.1 History of Audio/Video Networking

The idea of using packet networks –such as the Internet- to transport voice and video is not new. Early work on this issue started at 1970’ s and the first RFC (Request For Comment) was printed in 1977, the Network Voice Protocol (NVP) [7]. Studies on video transmission came later, however there is enough experience with audio/video conferencing and streaming on the Internet.

NVP was developed and tested for the ARPANET [8] predecessor of the Internet. A streaming service that is analogous with TCP/IP was provided with the ARPANET; however it had too much packet delay. Therefore, another UDP/IP like packet service protocol was offered for multimedia applications. After serving the 3-Mbps Wideband in 1980’ s,not only the voice transmission but also the packet video development was enabled. For the sake of multimedia transmission, NVP-II and ST/ST-II [9] have been offered by the Internet communities.

Starting from the 1990s, the processing power and multimedia capabilities of workstations and PCs became sufficient enough to enable the simultaneous capture, compression, and playback of audio and video streams. In addition to these improvements, development of IP multicast served new opportunities for Internet multimedia. For video/audio conferencing RTP that is based on NVP and its related protocols like SAP [10], SIP [11], SDP [12], H.261 [13], H.323 [14] etc. proposed. M-Bone tools enabled the use of these protocols efficiently. Some well known applications like VIC [15] and RAT [16] have been developed for real-time multicast conferencing.

In parallel with the development of multicast conferencing, the World Wide Web revolution took place, bringing glossy content and public acceptance to the Internet. Since network bandwidth and end system capacity have reached to a satisfactory level, the trend for video and voice streaming over Internet have been increased dramatically. Especially by the development of some well-known applications like RealAudio, QuickTime etc. have enabled the dense usage of multimedia along web pages. As a result of the enforcements coming from the market, developers have come up with a new solution for on-demand streaming applications. It is known as Real Time Streaming Protocol (RTSP) [17] and closely resembles HTTP the most well-known protocol in the wired-world.

### 2.2.2 Terms and Concepts for IP Conferencing

The main types of conferencing on IP are:

a. *IP point-to-point conferencing*: Two end-points communicate in real-time over an IP connection.
b. *IP multipoint conferencing*: Three or more endpoints communicate in real-time over an IP connection.
c. *MP* Videoconference: Two-way real-time audio and visual communication between two or more points.

While reading this MS Thesis, reader will be familiar with some terms that are strongly related with IP conferencing. Some of these terms are;

i. ITU-T H.323: A complete suite of protocols that facilitate multimedia communication over IP packets. Incorporated within this standard are video codec such as H.261 and H.263, audio codec, such as G.711, G.723.
ii. IETF SIP: Session initialization protocol (SIP) is a signaling protocol that focuses on session initiation, modification and termination.
iii. IETF RTP/RTCP: Real-time transmission protocol and its control protocol are the common streaming protocols for multimedia data such as audio or video.

## *2.3 P2P Networking*

Actually, peer-to-peer (P2P) computing is not a new concept. "Someone can say that when two computers were first connected, they formed a P2P network. All client/server architectures like mail servers, network news servers and domain name servers operate in peer-to-peer networks” [26]. Although P2P networking has a long history and many practical examples exist in the Internet, the term P2P is relatively new and hot topic in the market.

“Terminologically, describing P2P is not a simple thing. Probably it may be best described by what it is not rather than trying to pin down what it is. First of all, P2P is not about eliminating servers. It is not a single technology, application or business model. Perhaps most controversial is that it should not be characterized strictly by degree of centralization versus decentralization” [26]. For instance, despite the fact that two analog programs Napster [18] and Gnutella [19] file-sharing applications designed on the concept of P2P, they have major differences; namely; Napster acted as a traditional client server when users were looking for music, and it acted as a P2P network when users transferred files. This means that the system took advantage of the fact that it was easy to create a centralized database of music files and their locations, but very difficult and costly to host the music. Unlike to Napster, Gnutella is a completely decentralized P2P network. System is completely distributed and all peers in the system are identical.

What has happened and changed so P2P has become a hot issue in the market? What are the reasons of this unprecedented rise? Answer is concealed in today’ s requirements and improvements. These factors include the explosion of connected devices, the rapid increase of affordable bandwidth, acceleration of computing power and larger storage capacities and the proliferation of information at the edges of the network [26].

The common characteristics of today’ s typical P2P systems include the most of the following [26];

- Peer nodes have awareness of other peer nodes
- Peers create a virtual network that abstracts the complexity of interconnecting peers despite firewalls, subnets, and lack of specific network services.

- Each node can act as both a client and a server. Main target is building complete-symmetric system.
- Design and build more reliable and scalable systems. The overall performance and reliability of the P2P applications tends to increase as more nodes are brought online as opposed to typical client-server environments where more clients degrade performance.

Although P2P concept proposes better solutions, as all the other good things P2P has come with its handicaps. Most known problems of P2P systems are end-to-end synchronization and bulk messaging. Since computing is distributed over the peers and the lack of a centralized controlling mechanism, it may be strongly possible to be trapped in deadlock or erroneous situations. To measure these problems architects define new additional messages for specific purposes generally resulted with bandwidth saturation.

Today P2P networking has been applied to several fields of computing; chat, collaboration and white boarding, games, file sharing, content distribution, multimedia streaming etc. are the applications which have examples for P2P implementations.

# 3 RELATED WORK

In this chapter some previously developed techniques for video conferencing are going to be described, briefly. Especially, multicast techniques and some of their (popular) applications are going to be examined. Specifically, Narada and NICE protocols are told in separate sections while the rest are told in a merged part.

## *3.1 Multicasting*

Multicasting is an efficient mechanism for packet delivery in one-to-many data transfer applications. It eliminates the packet redundancy in the network. It is a very useful technique since it also decreases the number of states kept at any receiver node for multiparty applications. Today most of the recent techniques for P2P media streaming on the Internet utilize multicast model at the application layer instead of network layer. The main benefit of implementing application layer multicast is overcoming the lack of large-scale IP multicast deployment [20]. Besides, there is no need to change the network infrastructure for supporting application-layer multicasting [21, 22]. Instead, application layer multicasting uses the resources of end-users for packet forwarding and it provides higher level features such as reliability, congestion control, flow control and security. Although application layer multicasting is simpler and more scalable than the network layer multicast, application layer multicast is less efficient than the native multicast, since the application layer multicast protocols must send the identical packets over the same link. Common drawbacks of using application layer multicast are increased delay and higher bandwidth usage. [23]

In addition to P2P models, there exist other application layer multicast solutions adapting an overlay-based (infrastructure level) approach. Overlay-based approaches

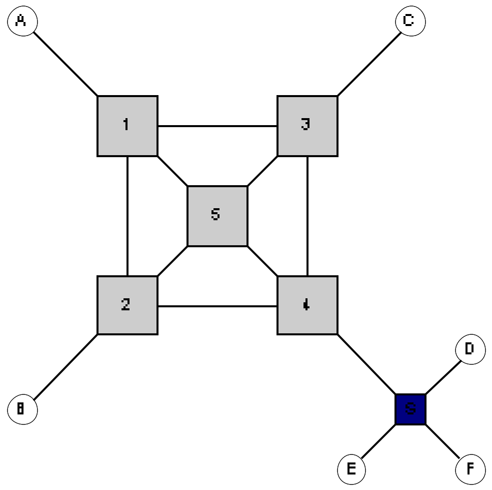


Figure 3.1.a: Architecture of the network. Small circles with letters are the end points. Rectangles filled with grey are the intermediate devices: router at figures b and c, and MSN at d. Small rectangle represent a dummy device like hub.

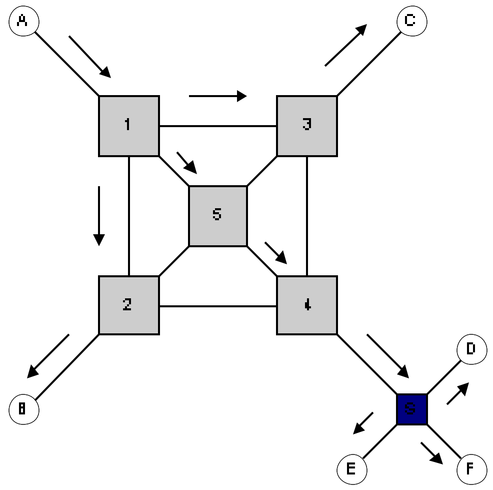


Figure 3.1.b: Network layer multicast
Data is scattered using router capabilities. A prepares one copy of the data and sends it to the router 1 and router 1 does the required multiplications. After router 4, data is received by a dummy device like hub and scattered to the connected members.

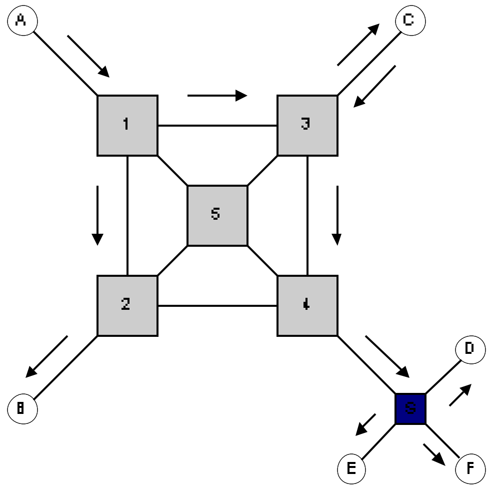


Figure 3.1.c: Application layer multicast
Data is routed at the end points. Here, A prepares one copy of data and sends it to B and C without using MBone infrastructure. Then C conveys the data to the rest of the user.

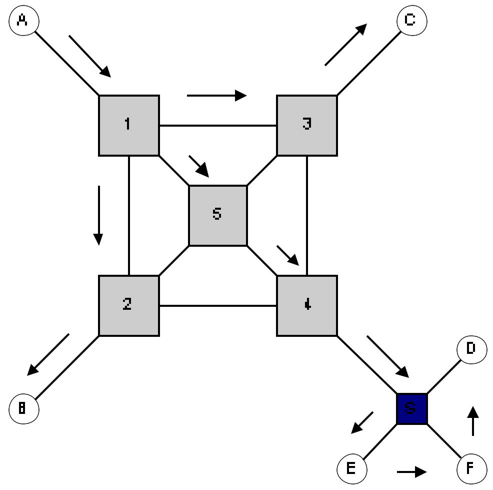


Figure 3.1.d: Overlay-based multicast
Routers are replaced with middleware (MSNs). In this example, MSN-1 makes a decision about using MBone infrastructure for B and C where the rest is notified using unicast communication.

**Figure 3. 1** Multicast architectures

require the existence of dedicated servers as opposed to P2P ones. They are scalable since the receivers can get the content from the dedicated servers acting as software routers, which would decrease the bandwidth consumption at the source as well. In [24], Banerjee considers a two-tier infrastructure to efficiently implement large-scale media streaming applications on the Internet. This infrastructure called the Overlay Multicast Network Infrastructure (OMNI) consists of a set of devices called Multicast Service Nodes (MSN) [25]. System requires user subscription to send multicast data service to the registered user. MSN makes decisions about how to scatter the data among the users. It shall use the network layer multicast or end-to-end (unicast) communication techniques. If unicast is the choice for data multicast then MSN has to calculate a feasible route for data transmission.

Figure 3.1 shows the architectures for the mentioned multicast techniques with simple examples. Figure 3.1.a displays the architecture of the example. There are six end users, and three of them are behind a dummy network device like hub and connected directly to themselves. Circle is used to represent the data receiver (end user) where rectangle is used to represent the intermediate devices/applications like routers or MSNs.  In figure 3.1.b network layer multicast is shown where user-A sends just one copy of data and this data is multiplied at the intermediate nodes (routers). Here user-A has a passive role for data delivery.  On the other hand, figure 3.1.c depicts a scenario where user A is totally responsible for data delivery. User-A calculates a route and prepares two copies of data and sends it to user-B and user-C. User-A also sends an additional message which notifies user-C about the next hop. Using this adjunct message user-C relays the incoming message to new destination(s). Finally, figure 3.1.d shows a case where routers are replaced with MSNs (a kind of middleware). MSNs are the central devices and are capable of calculating routing paths. Here, MSN-1 uses network layer multicast for user-B and user-C. Unlike user-B and user-C, rest of the users is delivered using unicast communication techniques. MSNs provide a way to build a hybrid architecture using network and application layer multicasts.

## *3.2 Protocols Based on Application Layer Multicasting*

### 3.2.1 NICE Protocol

NICE [24] is an application layer multicast protocol which has been developed in the University of Maryland. According to the creators of NICE, it is a recursive acronym for the Internet Cooperative Environment. NICE protocol is a one-to-many data stream protocol for low-bandwidth real-time applications. Although large receiver set increases control overhead and data latency for streaming applications, NICE makes it possible to develop data stream applications which have provably small (constant) control overheads and low latencies. NICE achieves this using a tree based solution for data multicast. Furthermore, it does not need to know about the underlying topology for data delivery paths.

NICE protocol arranges the set of end hosts into a hierarchy. This hierarchy requires grouping the end users at clusters and layers. The solution space which includes all enumerated end users is divided into several layers where each layer includes several clusters. Layers should be named as $L_i$ and i represent the order of the layer starting from the top. User nodes are grouped under clusters (group size is between k and 3k-1 where k is an external constant) at the layers and each cluster has a manager residing also at the upper layer. Managers are selected according to the distance value. In each group user who has the highest distance to the others is selected as the manager user. Responsibility of the manager user is to keep the states of the other members of the group and to decide how to deliver data among the group members. Therefore group members update their states at the manager user periodically. Naturally this increases bandwidth usage and causes temporary interruptions at the clusters.

Banerjee [24] offers an algorithm for adjusting clusters. Algorithm forces a host to be registered to one cluster at any layer. Second assumption of the algorithm refers that if user is present in some cluster in Layer $L_i$, then it must also occur in lower layers and it will be assigned as the cluster leader for the lower layers. On the other hand, if a host is not present in Layer $L_i$, it cannot be present in any layer $L_j$, where j>i. Algorithm has some limitations over the cluster sizes just to support balanced groups. Therefore each cluster is limited with a size of [k, 3k-1]. As a result, at any cluster, there are at most $\log_k N$ layers, and the highest layer where there is only one cluster being created with a single member. Figure 3.2.a shows a visual example of the NICE solution space. Here end users are represented with black nodes and their connection with (continuous) lines. Dotted rectangles are used to group the end users in clusters where dotted lines are used to relate the users at different layers.

NICE guarantees a constant control overhead O (logN) at the end users. On the other hand, it suffers from some basic operations like joining and leaving. For instance, join operation requires a hierarchical search starting from the highest layer. All clusters are queried (by managers) to select the most suitable one for joining. This burdens a cost of O (k logN) to the system. Figure 3.2.b tells a simple example for join operation. Arrows show the directions for the query messages. Due to its complexity and increased cost, an optimization for the join operation was offered. Soft-join/leave was offered as the optimization which provides a progressive approach for the whole system and is a well-known technique for suppressing the side affects. For this purpose, it is assumed that a user called Rendezvous Point (RP) is known by all other users. In NICE, this RP is used to prevent some soft-state cases and decrease the cost of join/leave operations. Using RP a new user does not need to be located to its final cluster at the beginning. It may reside at a temporary location and pass its cluster gradually.

Finally, drawbacks of the NICE protocol can be summarized in some titles;

- It allows single-source streaming where multi-source streaming should be provided in applications like video conferencing.
- It is (a little) complex to implement since layer and cluster management requires handling issues like security, maintenance etc.

- It is not fully-distributed. Thus it suffers from Single Point of Failures (SPF).

SPFs result with temporary unstableness. NICE has an election algorithm for choosing the cluster managers. If an SPF occurs then system will have faulty-states until election completes.

### 3.2.2 Narada Protocol

Narada [27] has been developed by considering some important objectives in mind. These objectives are self-organization, overlay efficiency, self-improvement and adaptation to the network dynamics. In Narada, end systems (users) self-organize into an overlay structure using a fully-distributed protocol. Further, end systems attempt to optimize the efficiency of the overlay by adapting to network dynamics and by considering application level performance. Unlike to its predecessors (e.g. NICE), Narada just targets the real-time applications like audio and video conferencing with average sized members.

In essence, Narada is based on the End System Multicast (ESM)[5]. ESM is a peer-to-peer video streaming system which supports high quality (300Kb-1Mb) video streaming over thousands of simultaneous users and developed at Carnegie Mellon University. ESM provides end-systems self organization. Using ESM, an end-user can monitor its statistics and adjust its order in the broadcast tree considering the performance aspects.

Narada protocol offers a tree based (minimum cost spanning tree) approach for data routing as in NICE protocol. The intuitive approach to constructing overlay spanning trees is to construct them directly—that is, members explicitly select their parents among the members they know [22]. Narada, however, constructs trees in a two-step process. First, it constructs a richer connected graph that is termed a mesh, and tries to ensure that the mesh has desirable performance properties (e.g. paths' delays and bandwidths) In the second step, Narada constructs spanning trees of the mesh, each tree rooted at the corresponding source using well known routing algorithms. Figure 3.3.b presents an

[5] see http://esm.cs.cmu.edu/

example mesh that Narada constructs for the physical topology shown in figure 3.3.a, along with a spanning tree rooted at A. Mesh-based approach has been used for two reasons; (i) it is motivated by the need to support multi-source applications and (ii) single shared trees are susceptible to a central point of failure, and are not optimized for an individual source. Therefore constructing a good mesh is an important topic at Narada. Two criteria are considered for mesh construction; (i) quality of the path between users (bandwidth, delay etc.) and (ii) maximum number of neighbors for a node at the mesh. Unfeasible connections which have high costs are cut from the mesh. Figure 3.3.c shows a visual topology example after the Narada protocol has been implemented.

At the data transmission side, Narada uses TCP Friendly Rate Control (TFRC) [28] as the underlying transport protocol on each overlay link. TFRC is rate controlled UDP and achieves TCP-friendly bandwidths. It does not suffer from delays associated with TCP such as retransmission delays and queuing delays at the sender buffer.

Finally, it is worth to talk about the some problems of Narada. They can be listed as below;

- Narada is optimized for only single sender scenarios [45].
- Narada might cause possible bandwidth bottlenecks since two different nodes in the mesh might be responsible for replicating the same data [45].
- As mentioned before, route calculation needs a two phase process;
  1st Step: is related with constructing a full mesh. To do something like this, nodes have to exchange their advertisements. Message exchanging is based on Distance Vector Routing (DVR) algorithm. Experiments have shown that these messages take a considerable percentage of the traffic. Protocol Overhead (ratio of non-data traffic to whole data traffic) calculated as;

  Group size = 128 members, Data Size = 16 Kbps, POR = 0.25

  Group size = 128 members, Data Size = 128 Kbps, POR = 0.04

  Group size = 128 members, Data Size = 64 Kbps, POR = 0.02

On the other hand, it is not possible to give full-quality service unless all nodes share the advertisements.

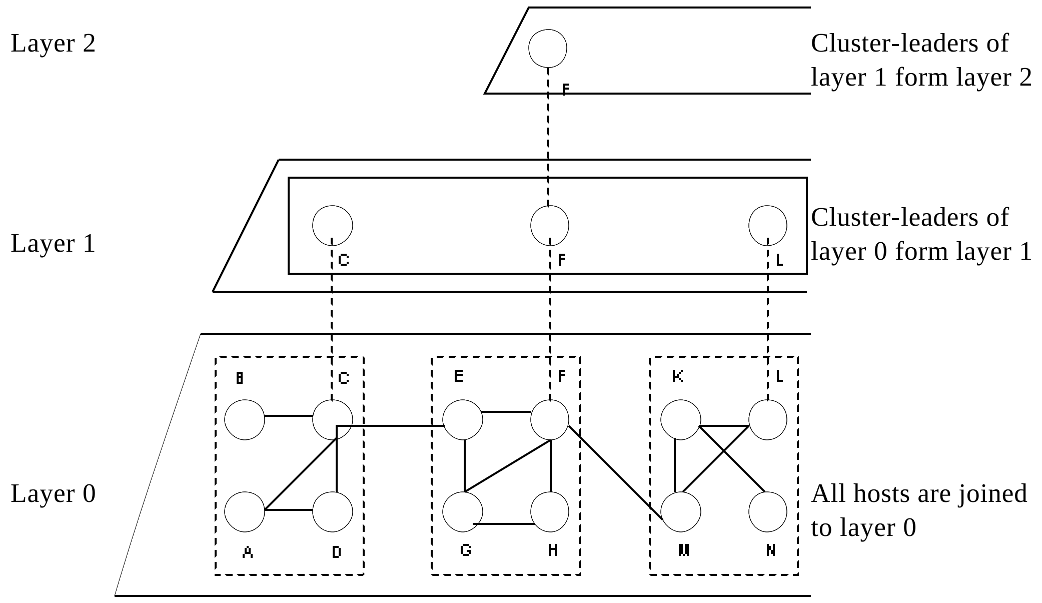
Unlike to Narada, Nebula does not require any state-info messages since each node’ s state is private to itself. By the way, chain construction mechanism is based on local user’ s status, not remote ones.

2nd Step: that starts after building the mesh, requires a mechanism to find a feasible path for data. Its cost is related with the Shortest Path Algorithms (SPFs). On the other hand, Nebula chain reconstruction is too easy and its cost is (in the worst case-demanded node is relayer-) 6 configuration messages. Length of the chain does not have any impact on the number of configuration messages.

As a final comment for Narada, constructing the mesh is a relatively slow process, and since the mesh is dynamic, changing tree links may affect the quality of video conference.

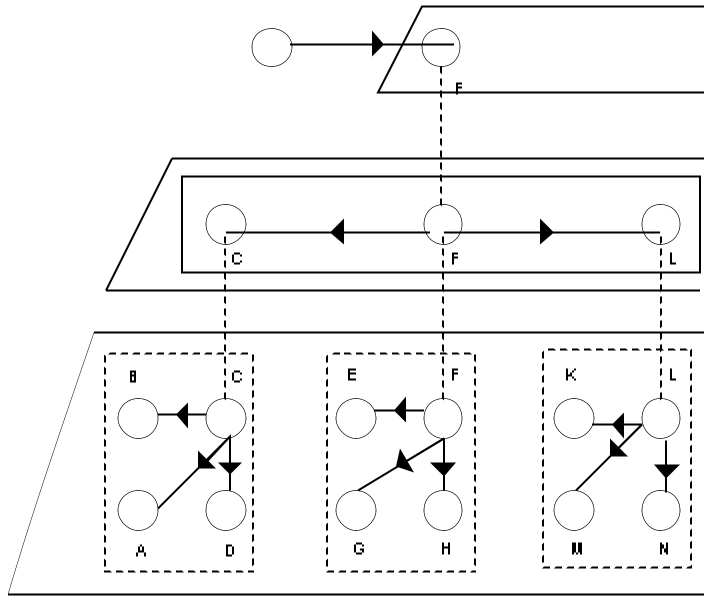
### 3.2.3 3D Video Conferencing

[29] Proposes an approach for multi-sender 3D videoconferencing using end-system multicast. It maintains a conference session protocol for small number of participants where each participant can act as video source for the rest, and frequent changes are handled quickly. It also offers a method of soft-join which involves locating a new participant into the multicast tree as fast as possible. Unlike video sharing, multicast tree construction in [29] is done via rendezvous points (RP) acting as central machines. This makes decision making fast and reliable. However, due to the nature of centralized solution, it suffers from single point of failures. In addition, multicast tree construction involves controlling all multicast trees which increases operational complexity and cost. While constructing multicast trees, the RP can choose one idle/available participant for acting as a reflector node (node that relays other participant’ s video without watching) without its consent.



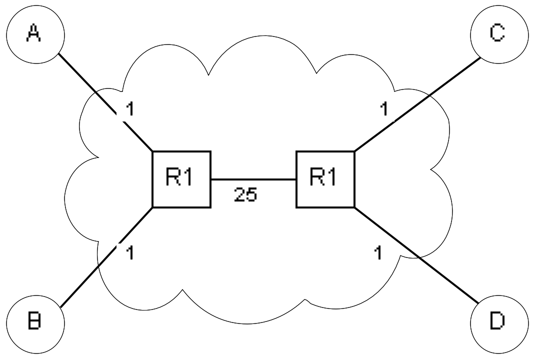
Layer 2
Cluster-leaders of layer 1 form layer 2
F
Layer 1
Cluster-leaders of layer 0 form layer 1
C
F
L
Layer 0
B
C
E
F
K
L
A
D
G
H
M
N
All hosts are joined to layer 0

.*a*) Example for NICE hierarchy. Black circles represent the end users and their connections are drawn with (continuous) lines. Dotted rectangles are used to group the end users in clusters where dotted lines are used to relate the users at different layers.

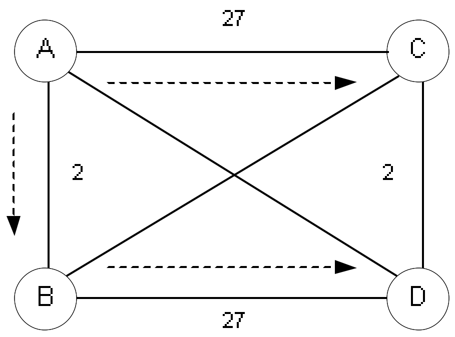


.*b)* Example for NICE join operation. User depicted with gray circle wants to register to system. It sends a registration message to the user resides at the highest layers. Then iteration starts for finding the suitable cluster for the new user. Each group manager queries its own group and returns the result back to the upper layer.

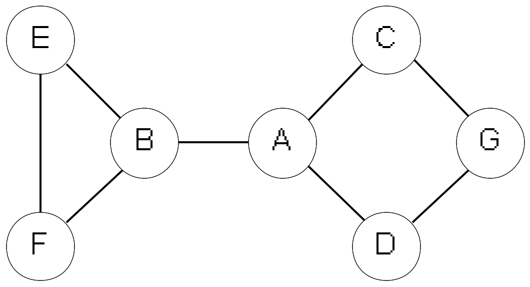
**Figure 3. 2** NICE protocol



*a*) Physical topology example



*b*) Example to illustrate the mesh-based approach in Narada



*c)* A sample Narada (overlay) topology

**Figure 3. 3** Narada Protocol

### 3.2.4 Other Related Work

ZIGZAG [30] is a P2P system developed for single-source media streaming to a large group of participants on the Internet. It proposes an efficient failure recovery and control protocol for maintaining the application layer multicast tree against network dynamics and unpredictable client behaviors. Failure recovery is performed locally, and control overhead is constant. In addition, the system addresses the issue of minimizing end-to-end delay from source to receivers. The study includes a theoretical analysis together with a simulation study and comparison with NICE protocol. NICE and ZIGZAG are different in their multicast tree construction and maintenance mechanisms. They both focus on large P2P networks. The system [30] is based on one of the first P2P streaming techniques called "chaining" [31]. It allows a source to server chain of clients using a single data stream. In chaining, the delivery tree is built rooted at the source, and it uses the source's bandwidth efficiently by utilizing other peers' bandwidth to stream media to others.

The main benefit of implementing application layer multicast is overcoming the lack of large-scale IP multicast deployment. CoopNet [32] which is also an application layer multicast protocol, proposes a hybrid approach integrating the client-server and P2P models for both live and on-demand streaming media. For on-demand media, clients make use of caching content that they viewed recently. For live media streaming, clients form a distribution tree rooted at the source. In particular, CoopNet considers the problem of server overload by a large number of requests from clients. In such a case, it utilizes P2P model and clients cooperate to distribute content to decrease the load on the server. In addition, CoopNet uses a multiple description coding method for the media content, which improves robustness and balances load among peers.

# 4 NeViC: ARCHITECTURE & IMPLEMENTATION DETAILS

## *4.1 Introduction*

This thesis proposes a new solution which differs from the prior work discussed in the previous chapter in a number of ways. First of all, the solution is not based on an application layer multicast model or overlay-based approaches requiring the availability of dedicated servers. The solution employs a fully distributed model utilizing peer-to-peer principles essentially for the dissemination of media content. Hence, problems of single point of failure and server overload are avoided. As discussed in the following sections, main assumption (of the thesis) dictates that each conference participant can produce, receive and send/relay only one video signal at any given time. In other words, only one video signal can be used for both incoming and outgoing channels. This limitation is called as "One-to-one I/O Restriction". This, together with the peer-to-peer media dissemination model, leads to foreseeable and balanced overhead at participants. As it will be seen later, the solution forces the participants to utilize the bandwidth consumption. Besides, it does not allow resource sharing without a participant' sconsent. Proposed solution achieves these goals via the "chaining" methodology.

The proposed solution can be used to implement a peer-to-peer multi-point videoconferencing system that can configure itself to allow each participant to see any other participant whenever possible i.e., when the resulting configuration is achievable. In this architecture, it is assumed that the usual tasks of setting and controlling a multipoint conference are handled by the appropriate protocols as described in the

Internet multimedia architecture [33]. Therefore, tasks such as inviting participants to conferences, distributing and managing the list of active participants, etc., are outside the scope of this description. Also, transmission of any other media associated with the video such as text messages is not addressed. These may or may not use a peer-to-peer system.

This chapter begins with examining the details of the fully-distributed solution for multi-point video conferencing. Both architecture and implementation details are described. On the other hand, this chapter also includes not only the peer-to-peer solution narration but also a technique for the solution verification.

## *4.2 Chain Based Solution*

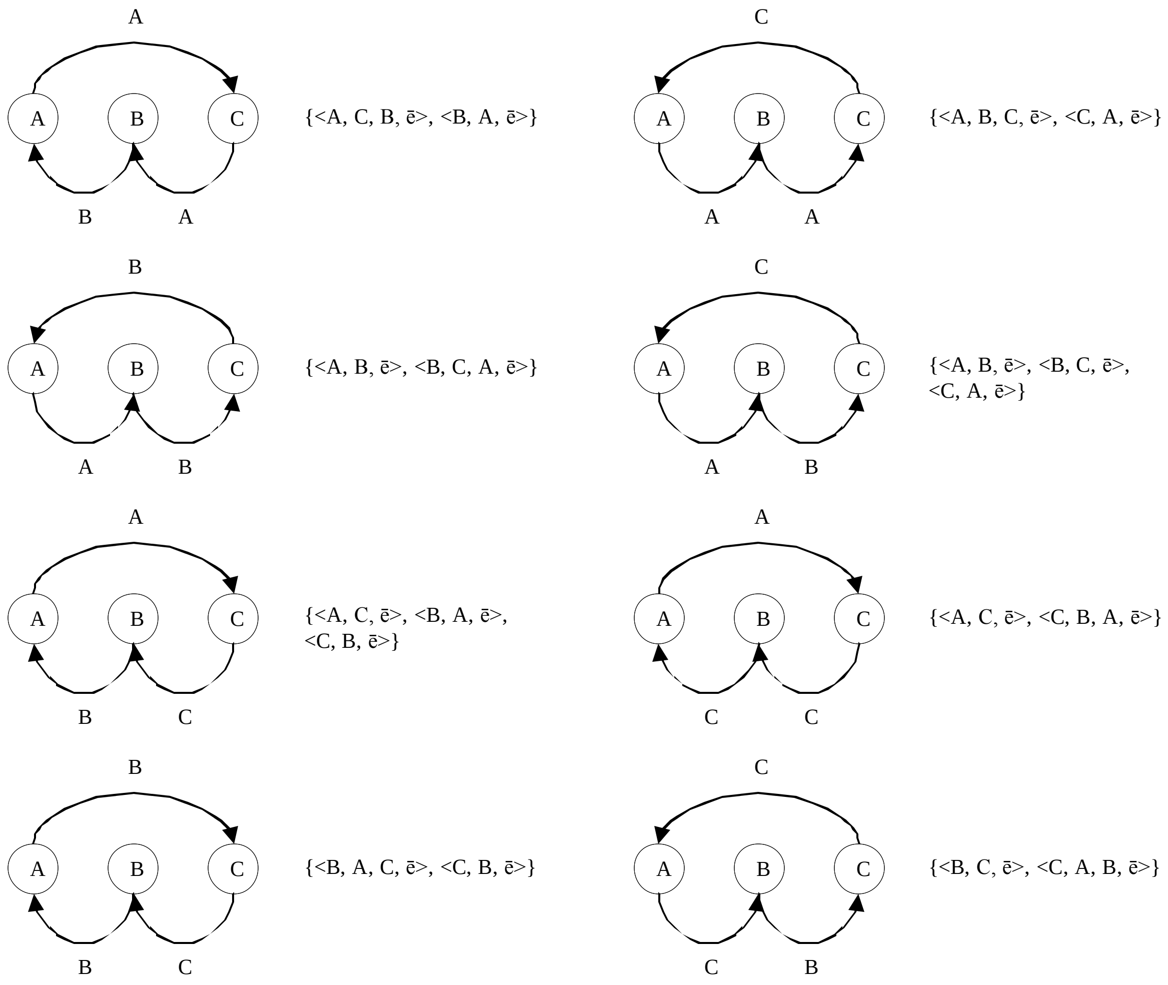
The state of the peer-to-peer multi-point videoconferencing system is defined via a concept called "chain". A chain defines the video flow configuration between the participants of a conference receiving the same video signal at any given time. All the chains involved in a peer-to-peer multi-point videoconference define the conference's current state. Each participant's chain info is private for itself and it cannot be reached by the other participants registered for the conference. The video flow configurations, i.e. chains, change during a session as a response to "configuration messages" sent between the participants or when a participant leaves a session with or without informing anyone, where the latter may happen because of system or participant crashes, etc.

A chain is an ordered list of identification numbers of the conference participants who receive the same video signal, in the order they receive it. The first entry in this list, called the "head", is the video source for the rest of the participants in the list. Every participant gets assigned a unique identification (ID) number when they join to the conference. Each chain has an additional binary valued attribute called "extensibility". A chain is extensible (e) if its last member is not a head for another chain (means output channel is idle), non-extensible (means output is busy) (ē) otherwise.

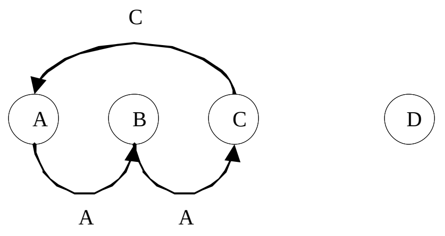
Each chain member knows the head of the chain and the receiver ID if it relays video. The head knows about the entire chain. A participant who does not send or receive video is a member of an extensible chain containing only its own ID number. For an N-user conference session, there are at most O ($N^N$) different chains ignoring the ordering problem. That means, only the participants’ receiving states are considered while calculating the complexity. For instance, if there are 3 users registered for the conference, there will be $3^3$ (27) different states present. For each user, there are 3 different receiver states that contain the two other users and idle state. Namely, user-A may receive from user-B or user-C or nobody at any given time.

In order to enable multi-point video; however, a participant receiving a video signal may have to pass (relay) it to another participant who also wants to receive the same video signal. Relaying the received video signal to another participant does not bring any additional computational burden beyond sending the participant’ s own video signal, but it uses up the entire available upstream bandwidth. Thus, a participant who is sending its own video signal can’ t act as an intermediate peer passing a video signalto another peer, and, an intermediate peer can’ t send its own video signal to anyone else.

Figure 4.1.a includes the chain configurations both in visual form and list notation. As shown in figure 4.1.a, in a MP videoconference among three participants, a peer-to-peer configuration can always be found such that any participant can see any other participant at any given time. On the other hand, as demonstrated in figure 4.1.b for a conference with four participants; however, there are cases where this will not be possible when the number of participants is four or larger. These unavailable states are called “deadlock” states where a request cannot be satisfied since current system does not have a valid solution.



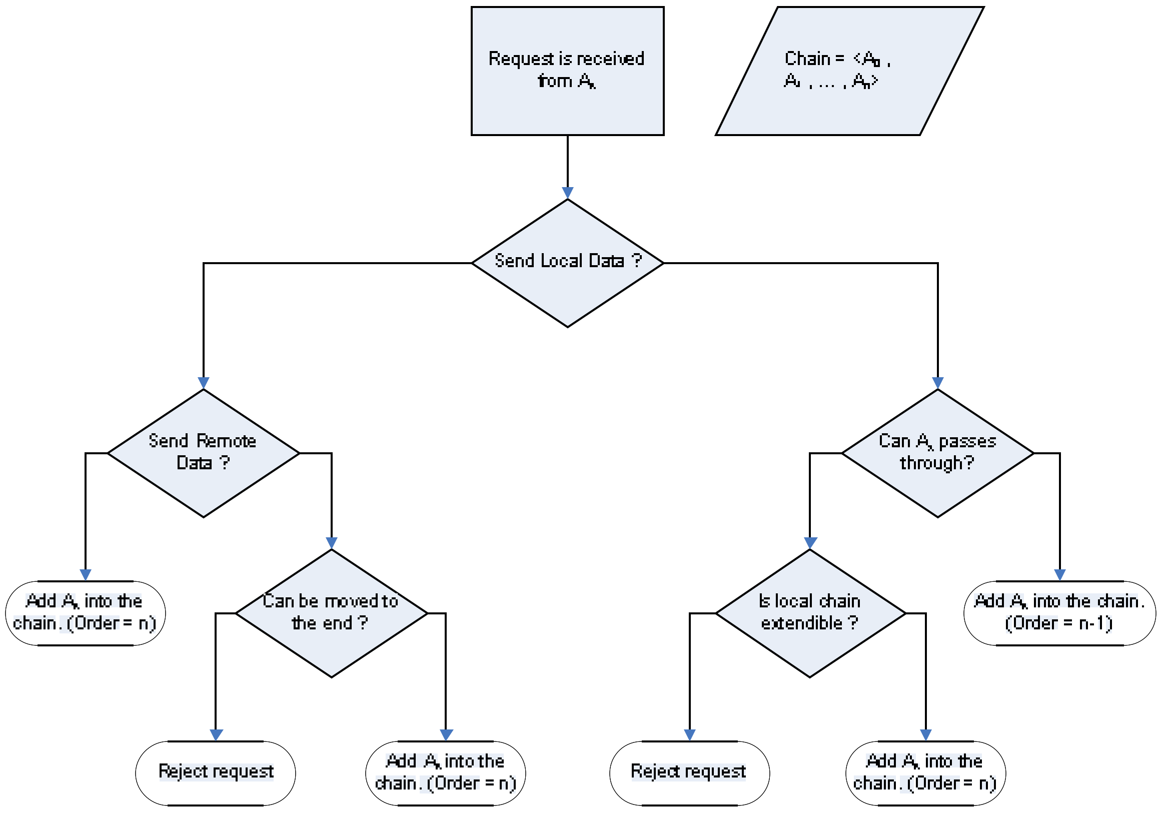
.*a*) Chain examples



User D requests the user B in the conference. However, there is no way to send B’ s video to user D

.*b*) Deadlock Example

**Figure 4. 1** Participant chain


Request is received from $A_x$
Chain = $<A_0, A_1, \ldots, A_n>$
Send Local Data ?
Send Remote Data ?
Can $A_x$ passes through?
Add $A_x$ into the chain. (Order = n)
Can be moved to the end ?
Is local chain extendible ?
Add $A_x$ into the chain. (Order = n-1)
Reject request
Add $A_x$ into the chain. (Order = n)
Reject request
Add $A_x$ into the chain. (Order = n)

```
Request is received from $A_x$
        Control if local video is sent {
                Control if $A_x$' s output is idle {
                        Add $A_x$ into the chain before the last member.
                }
                Else {
                        Control if local chain is extendible {
                                Add $A_x$ into the chain as the last member.
                        }
                        Else {
                                Reject Request.
                        }
                }
        }
        Else {
                Control if remote video is relayed {
                        Control if peer can be assigned as the last member at its chain {
                                Add $A_x$ into the chain as the last member.
                        }
                        Else {
                                Reject request.
                        }
                }
                Else {
                        Add $A_x$ into the chain as the last member.
                }

        }
```

**Figure 4. 2** Chain construction algorithm

Whenever a user receives a request from the remote host, it has to make a decision using the algorithm given in figure 4.2. Both the flow-chart and pseudo-code of the algorithm are displayed. A user has five different states at any given time. These states and their behaviors are summarized as follows.

- *Idle*: All channels are idle. No data transmission is available.
- *Receive*: Only the incoming channel is busy. Video is being received and delivered to the display module.
- *Send*: Only the outgoing channel is busy. Video is being produced and sent to the remote user.
- *Receive and Send*: Both the incoming and outgoing channels are busy. Video is being received (remote packets) and sent (local packets) over transmission channels.
- *Receive and Relay*: Both the incoming and outgoing channels are busy. Video is being received (remote packets) and sent (remote packets) over transmission channels.

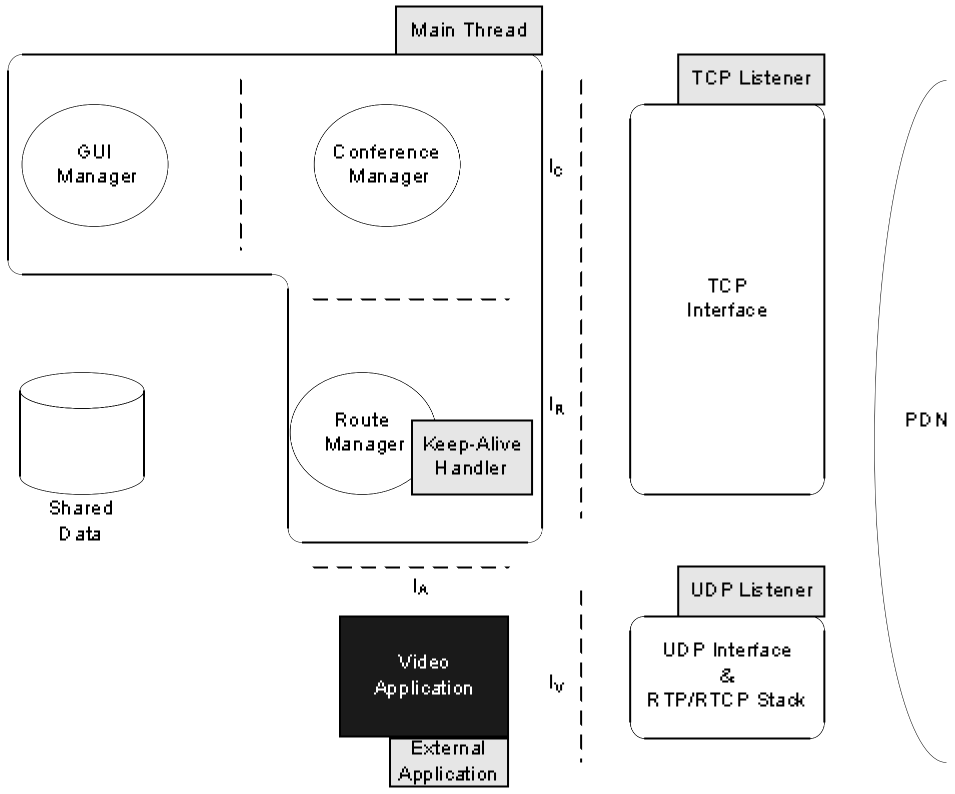
According to the algorithm in figure 4.2, there are totally 6 different paths where two of them result with unsuccessful attempts. If a user is currently in receiving and relay state and the header chain is not extensible, then there is no way to ensure video transmission for the demander user. Other unsolvable case occurs if the demander' s output channel is busy while the requested user' s channel is not extensible. Other cases are handled by the algorithm. Conclusion part includes a test case which evaluates the efficiency (ability to find a solution) of the algorithm.

## *4.3 NeViC Architecture*

In order to see the behavior and efficiency of the proposed solution, a complete application called Nebula Video Conferencing (NeViC) has been designed and developed. NeViC has six main modules as demonstrated in figure. 4.3. The first module is for conference management. A standard conference management application can be used for this purpose. In the application, I have developed a simple conference management application with its basic functionality such as conference creation and joining in and leaving a conference. This module has three types of messages: (i) *invitation message* is used to notify remote users about a conference that a peer participates in or creates, (ii) *login and logout messages* are used to register to and to leave a conference, (iii) and a *keep-alive message* sent periodically is used to trace other conference participants for detecting a possible user crash.

Another responsibility for this module specific to the application is to establish a unique session id for each conference participant. Each user identifies itself with a couple integer values "*<user-id, user-key>*". Both values are randomly generated and used for different aims. *User-id* field is the actual parameter used for authorization. On the other side, *user-key* field is used to authenticate the remote user since these keys are only known by the conference participants. So the risk of possible spoofing and Denial of Service (DoS) attacks are minimized. Nevertheless, this thesis does not target to find a solution for these topics.

Conference manager is tightly coupled with routing manager and TCP interface. For the routing manager, it listens for the current states of the remote users. In other words, if a user logouts or crashes, then conference manager notifies the routing manager about the case and enables it to take the required action or vice versa. The interface between the conference manager and the route manager modules is a non-standard one and implemented using shared memory. Actually, NeViC uses shared memory for the threads running on the same virtual machine (VM) and uses sockets for external applications. On the other hand, TCP Interface supplies configuration messages and traces the logs of each connection for the conference manager. Besides, conference manager has a direct relation with the GUI manager since nearly all user-based operations like login, logout, invite etc. are triggered by using a command panel.



Modules:

*GUI Manager*: Responsible for visual components. Run by Main Thread and TCP Listener

*Conference Manager*: Responsible for conference management. Run by Main Thread and TCP Listener

*Route Manager*: Responsible for chain construction and optimization. Run by Main Thread and TCP Listener

*TCP Interface*: Responsible for TCP message handling. Run by TCP Listener

*UDP Interface*: Responsible for UDP video packets. Run by UDP Listener

Interfaces:

$I_C$: Interface for Conference Management
$I_R$: Interface for Route Management
$I_A$: Interface for Video Application
$I_V$: Interface for Video Management

Applications:

*Video Application*: Application used to capture and display video packets. Application also performs coding related operations.

**Figure 4. 3** NeViC system architecture

Conference manager is not created as a separate process or thread. Its functionalities are executed whenever they are required by the Main and TCP Listener threads. All GUI functionalities and some networks operations are handled by the Main thread where the TCP Listener thread is totally responsible for incoming (network received requests) requests.

The second module, Route manager, communicates with the conference manager to send current chain status (to be displayed on the screen), and specific events like user login/logout and peer crashes. Its main responsibility is to provide valid and efficient chain construction mechanisms. It tries to find a way how to multicast local video. For this purpose, it includes a unit to construct multicast chains or trees. Initially, conference manager informs the route manager just about the local user-id. During the application period, route manager is only interested in user identification numbers. It keeps chains in a comma separated string format. “A, B, C, D” is a sample chain representation where the alpha-numeric characters must be replaced with valid integer values in the real-application. In this chain representation, first element shows the video source (header user) followed by the receiver ones. An important point here is that header user knows the other users’ states except the last one. Therefore if it is required, header user has to query the last user’s up-to-date state.

Finding a way for video multicast locally is not enough hence route manager does the remote updates by sending configuration messages. Tracing for possible user crashes, route manager runs a separate and independent thread Keep-alive Listener. Keep-alive messages are sent for a $T_S$ time period and checked for every $T_R$ timestamps where $T_R$ is calculated as a linear function of $T_S$ value. Equation is $T_R = n \times T_S$. n value is optional and used to determine max. time for keep-alive delivery. If the route manager detects a crashed participant, then it performs the following actions; (i) if it is the head node, then it calculates a new multicast chain and does the required changes on the chain, (ii) otherwise, it just notifies the head node of the chain about the crashed peer. Route manager has an additional profound role which will be described later. It runs the delay

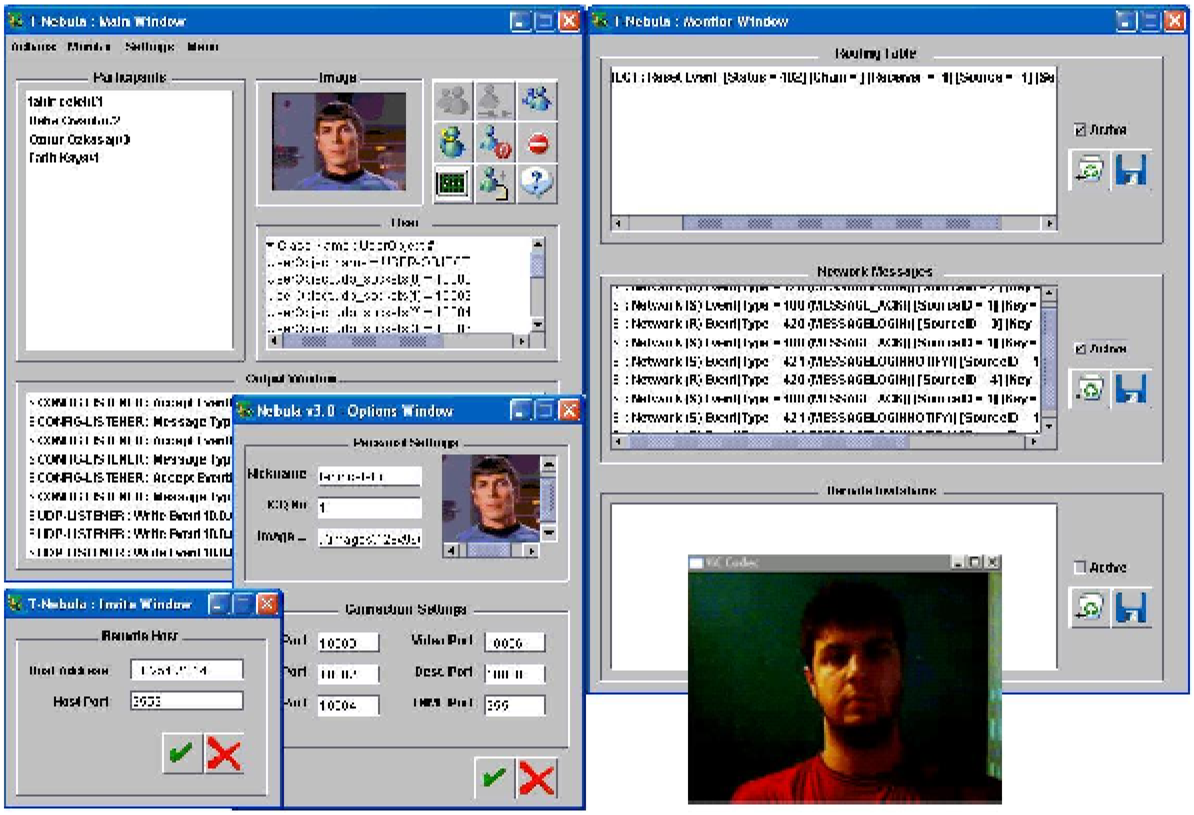


**Figure 4. 4** NeViC GUI

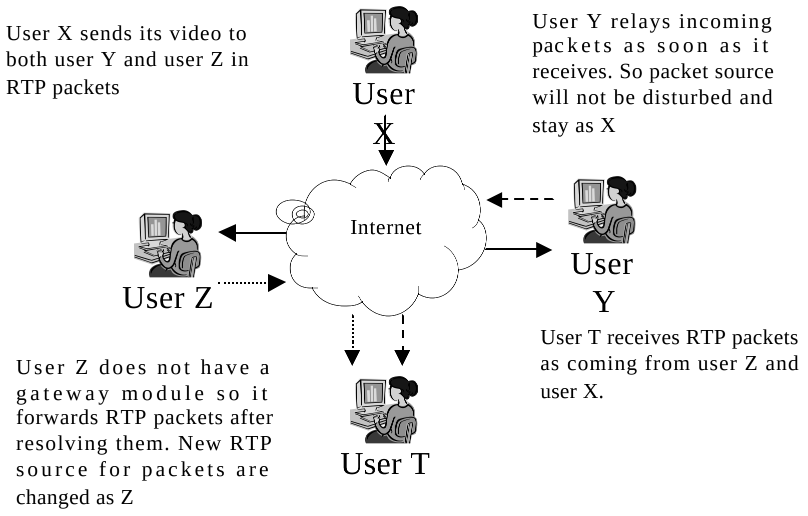

User X sends its video to both user Y and user Z in RTP packets
User X
User Y relays incoming packets as soon as it receives. So packet source will not be disturbed and stay as X
Internet
User Z
User Y
User T receives RTP packets as coming from user Z and user X.
User Z does not have a gateway module so it forwards RTP packets after resolving them. New RTP source for packets are changed as Z
User T

**Figure 4. 5** Gateway Example

measurement unit which has a vital importance for the chain optimization process. Similar to the conference manager, route manager is also run by the Main and TCP Listener threads. The only difference from the conference manager implementation is that, configuration messages are directly addressed to the routing module.

A user interacts with the system using the GUI module which may be implemented as an integral part of the conference manager. Video requests from other participants are given to the system through the GUI. Figure 4.4 displays the designed windows and dialogs together.

The video application handles video capture, compression/decompression, display and transmission over the network. It is a separate process and communicates with the NeViC application using socket interface. For this purpose, I have used a proprietary implementation of an H.263 based video codec. This system is designed to operate over a best effort network, and hence can gracefully handle stops in its input video bit streams and changes of the video sources during a session without a need to rapidly stop and restart the codec that may be problematic for many video codec systems. After porting the gateway module to the application, network transmission duty of video device is left to the gateway application. Video device just sends and receives video packets to\from gateway module directly without tunneling in RTP/RTCP packets. Tunneling of video packets in RTP/RTCP packets is the duty of the gateway application.

NeViC video application samples about 3-5 frames per second. It captures video frames in I420 which is a YUV format. In order to encode/decode video frames an open-source implementation called FFMPEG [34] has been integrated into the video capture/display application. Main strategy for encoding/decoding is using I and P frames. B frames are discarded. Therefore gop-size is set to 0 for the codec. For the sake of simplicity and packet independency, P frames are produced at most for every three video samples. To support packet-based transmission MPEG-4 [35] codec has been used. Since video packets are small enough, keeping P frames number small does not affect the application performance. Actually, it makes everything a little simpler. It eliminates the bulk P frames originated because of video interruptions. Otherwise codec has to be adjusted every-time needed for I and P frame orders.

Video device is configured by the route video manager to stop/start capture/display video using a simple interface. Route manager does this according to the user state described before. Video application prints local video samples on the screen unless incoming transmission channel is in use. On the other hand, output transmission channel does not have an affect on the display mechanism.

UDP Gateway module is a simple socket based process which just sends\receives packets to\from pre-determined UDP sockets. If a packet comes from a local UDP point, then this module sends it to the destination automatically. Otherwise, if a packet comes from a remote host, then gateway may re-send (relay) it to another destination and pass it to inner modules. Why do I need to detach network transmission part from the video device? Figure 4.5 describes the reason in a very simple fashion. After installing the RTP/RTCP stack, prototype transmits the video in RTP packets and video flow is controlled via RTCP messages. Assume that user X sends RTP packets to user Y (has gateway application) and user Z (does not have installed gateway application). Both receivers relay the incoming data to user T. As soon as user Y receives the RTP packets, it retransmits them without delay. RTP packets are not altered, they are just conveyed. Unlike user Y, user Z has no installed gateway application so RTP packets are directly gathered in RTP/RTCP stack. Thus, the retransmission of the incoming data is the responsibility of video device. Since new RTP packets are generated, RTP source will be changed for the relayed data. As a result, user T thinks that incoming data belong to user Z and user X. This erroneous case can be resolved with installing gateway application to user Z, as well. In conclusion, gateway application prevents source confusion stemmed from relay operation. This case is also valid for the RTCP messaging. In NeViC, RTCP messages are not activated but might be implemented as described in [36]. Technique for gathering RTCP messages described in [36] forces each user to send its RTCP messages directly to the destination. An integral solution is being rejected since it leads to big RTCP messages with higher delays.

## *4.4 NeViC Messages*

Configuration messages form an important part for both conference and route managers. It is important for conference manager since informal conference management protocol has been enabled by some simple messages. Specifically route manager also receives/sends related requests/responses via configuration messages. For example, if a new participant wants to receive the video from a head, the head needs to configure its chain and send the necessary messages to the affected participants. If a chain member detects loss of video, it waits for an "update" message from the head and if such a message does not arrive within a timeout period, the head is assumed to be non-functional and the chain is dissolved. NeViC generally uses a two-pass messaging strategy where the first message is for request and the second one is for response. Configuration messages are transmitted as objects on the wire. Java is capable of transmitting data as several objects and since (great deal of) NeViC has been developed using Java technologies; it uses a MessageObject.class for data transmission over TCP sockets. Declaration part of the MessageObject.java can be seen in the appendix part of the thesis (see the disc).

There are 9 configuration messages as outlined below;

*i)* *Login Message:* Figure 4.6 shows the communication required for login operation. Login operation is composed of two phases and it needs 4-way handshaking mechanism. In the first phase, (unregistered) remote user applies for conference registration. This request can be processed by any of the registered users. Second phase is related with notifying the rest of the conference members.

*ii)* *Logout Message:* Figure 4.6 shows the steps for unregistering a user from the conference. In order to define a new message type for user crashes, in logout messages values parameter is activated and filled with the identifiers of the unregistering users. Therefore, in all cases, a user has to fill values [1] field with the logout or crashed user's identifier.

*iii)* *Invite Message:* This message is used to invite the remote user to the current conference. In addition to sending the invitation, this step also performs the registration of the remote user if remote user accepts the invitation. Figure 4.7 describes the whole process.

*iv)* *Keep-Alive Message:* Used to determine if a participant is still in the session. Actually, this is among the duties of the conference management protocol. However, getting this information from the conference manager may take too long, resulting in a prolonged loss of video for several members. A participant starts sending "keep-alive" messages to its recipient as soon as it joins to a conference and this interval is adjusted as hard-coded. Recipient selection is done locally so there is no need for a selection algorithm. Each user chooses the next entry from the "active members" table by applying a modulus calculation. Since members are sorted according to their identifiers, it is guaranteed that possible confusions will be eliminated from the system. If a participant does not receive a "keep-alive" message for a pre-determined time period, it notifies each member separately about the case for taking its own action. Figure 4.7 shows the content of a keep-alive message.

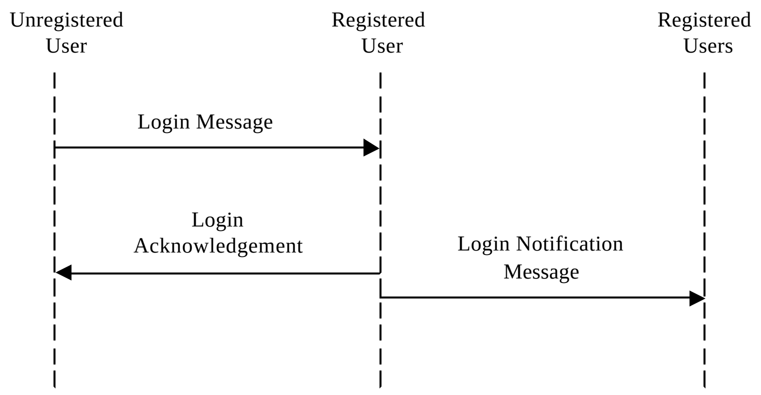
*v)* *Update Message:* Sent by the head to only one participant at a time to change its video receiver. This message includes the new receiver's ID.

*vi)* *Query Message:* A head sends this message to the last member of the chain to determine if the chain is extensible. This message is analogous to the update message since if the last member of the chain is extensible then its update can be done immediately without sending an additional "update" message. Figure 4.8 displays both update and query messages together.

*vii)* *Rebuild Message:* This message is generated by a participant (e.g. user-A) if it is in receive-relay state and it has a video request waiting. User-A sends rebuild message to its own head (user-B) to be moved to the chain rear. If user-B succeeds in assigning user-A as the last one, then user-A's output channel becomes idle and user-A can reply the video request positively. See figure 4.9.

*viii)* *Release Message:* When a participant decides to stop receiving, it must send a “release” message to its head. If the “release” message is not from the last chain member, the head reconfigures its chain by sending an “update” message to the member sending video to the source of the “release” message. This message includes the ID of the successor if any. If the last participant wants to stop video, the peer before the last is sent a “modify” message with a zero argument. See figure 4.9.

*ix)* *Request Message:* Sent by any participant to request video from another participant. The recipient may already be the head of a chain, in which case, the chain needs to be reconfigured if possible. Otherwise, a new chain can be created. A video request message carries the ID of the requestor and an attribute indicating whether the requestor is sourcing video to any other participant or not. A participant already receiving video can’t send a video request message. Various actions resulting from a video request message are outlined in figure. 4.2. See figure 4.10 for request message content.



*Login Message*

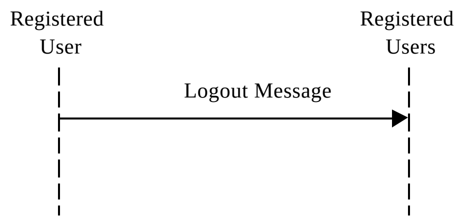
| | |
|---|---|
| Message.mint | INTERFACEMANAGEMENT |
| Message.type | MESSAGELOGIN |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | null |
| Message.param | user |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |

*Login Acknowledgement*

| | |
|---|---|
| Message.mint | INTERFACEMANAGEMENT |
| Message.type | MESSAGEACK |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | [0] = {OK, REJECT} |
| Message.param | users \| null |
| Message.xparam | <conference alias> \| null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |

*Login Notification Message*

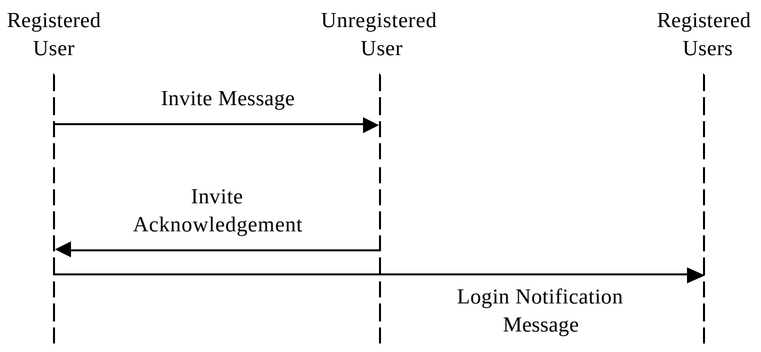
| | |
|---|---|
| Message.mint | INTERFACEMANAGEMENT |
| Message.type | MESSAGELOGINNOTIFY |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | null |
| Message.param | user |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |



*Logout Message*

| | |
|---|---|
| Message.mint | INTERFACEMANAGEMENT |
| Message.type | MESSAGELOGOUT |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | [1] = <user id> |
| Message.param | null |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |

**Figure 4. 6** Login and logout message descriptions



*Invite Message*

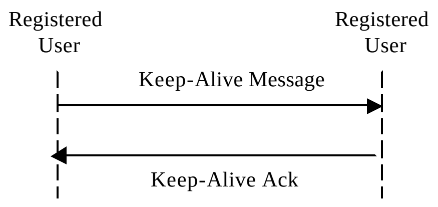
| | |
|---|---|
| Message.mint | INTERFACEMANAGEMENT |
| Message.type | MESSAGEINVITE |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | null |
| Message.param | users |
| Message.xparam | <conference alias> |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |

*Invite Acknowledgement*

| | |
|---|---|
| Message.mint | INTERFACEMANAGEMENT |
| Message.type | MESSAGEACK |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | [0] = {OK, REJECT, BUSY} |
| Message.param | user \|\| null |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |

*Login Notification Message*

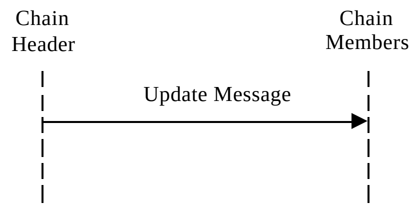
| | |
|---|---|
| Message.mint | INTERFACEMANAGEMENT |
| Message.type | MESSAGELOGINNOTIFY |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | null |
| Message.param | user |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |



*Keep-Alive Message*

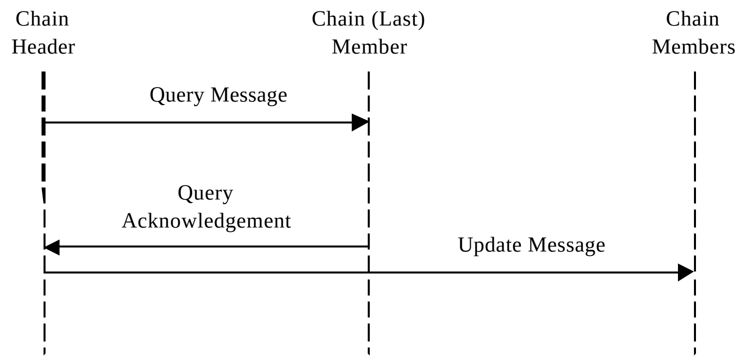
| | |
|---|---|
| Message.mint | INTERFACEMANAGEMENT |
| Message.type | MESSAGEKEEPALIVE |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | null |
| Message.param | null |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |

**Figure 4. 7** Login notification and keep-alive message descriptions



*Update Message*

| | |
|---|---|
| Message.mint | INTERFACEROUTING |
| Message.type | MESSAGEUPDATE |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | [1] = <receiver id>, [2] = <sender id> |
| Message.param | null |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |



*Query Message*

| | |
|---|---|
| Message.mint | INTERFACEROUTING |
| Message.type | MESSAGEQUERY |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | [1] = <receiver id>, [2] = <sender id> |
| Message.param | null |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |

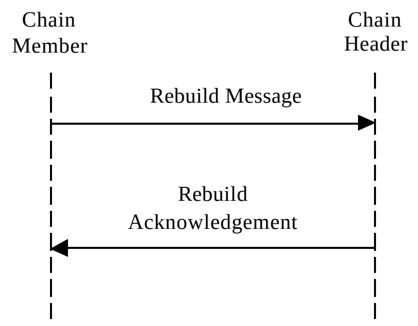
*Query Acknowledgement*

| | |
|---|---|
| Message.mint | INTERFACEROUTING |
| Message.type | MESSAGEACK |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | [0] = {OK, REJECT, BUSY} |
| Message.param | null |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |

*Update Message*

| | |
|---|---|
| Message.mint | INTERFACEROUTING |
| Message.type | MESSAGEUPDATE |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | [1] = <receiver id>, [2] = <sender id> |
| Message.param | null |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |

**Figure 4. 8** Update and query message descriptions

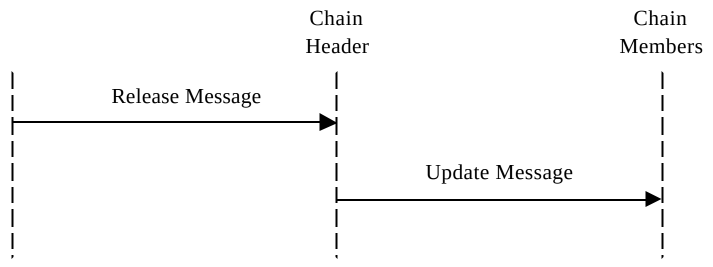


*Rebuild Message*

| | |
|---|---|
| Message.mint | INTERFACEROUTING |
| Message.type | MESSAGEREBUILD |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | null |
| Message.param | null |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |

*Rebuild Acknowledgement*

| | |
|---|---|
| Message.mint | INTERFACEROUTING |
| Message.type | MESSAGEACK |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | [0] = {OK, REJECT, BUSY} |
| Message.param | null |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |



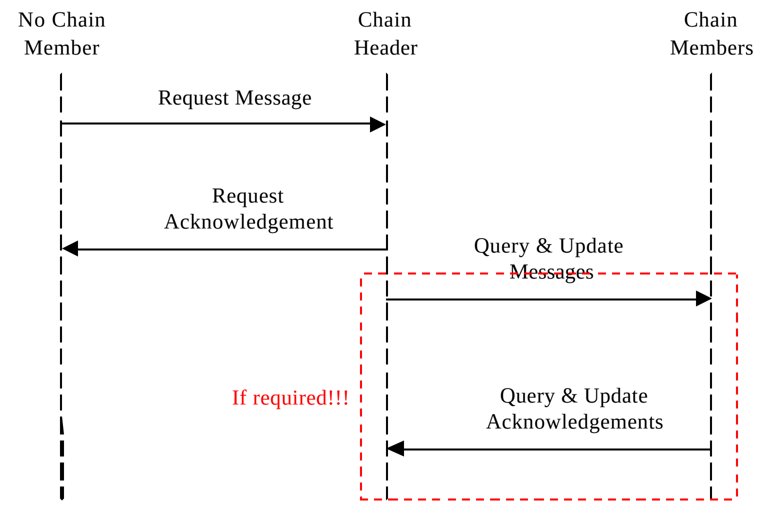
*Release Message*

| | |
|---|---|
| Message.mint | INTERFACEROUTING |
| Message.type | MESSAGERELEASE |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | null |
| Message.param | null |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |

*Update Message*

| | |
|---|---|
| Message.mint | INTERFACEROUTING |
| Message.type | MESSAGEUPDATE |
| Message.source | <message source> |
| Message.key | <message key> |
| Message.values | [1] = <receiver id>, [2] = <sender id> |
| Message.param | null |
| Message.xparam | null |
| Message.cdate | <creation time> |
| Message.stime | <set at sender> |
| Message.rtime | <set at receiver> |

**Figure 4. 9** Rebuild and release message descriptions



| *Request Message* | | *Request Acknowledgement* | |
|---|---|---|---|
| Message.mint | INTERFACEMANAGEMENT | Message.mint | INTERFACEMANAGEMENT |
| Message.type | MESSAGEREQUEST | Message.type | MESSAGEACK |
| Message.source | <message source> | Message.source | <message source> |
| Message.key | <message key> | Message.key | <message key> |
| Message.values | null | Message.values | [0],[1],[2] will be set |
| Message.param | [3] = <ct. state> | Message.param | null |
| Message.xparam | null | Message.xparam | null |
| Message.cdate | <creation time> | Message.cdate | <creation time> |
| Message.stime | <set at sender> | Message.stime | <set at sender> |
| Message.rtime | <set at receiver> | Message.rtime | <set at receiver> |

**Figure 4. 10** Request message description

```
line 0   ---------------------MODULE HourClock---------------------
line 1   |                                                        |
line 2      EXTENDS Naturals
line 3      VARIABLE hr              (* Variable declaration *)
line 4     HC_ini == hr \in (1..12)
line 5      HC_nxt == hr'  = IF hr #12 THEN hr + 1 ELSE 1
line 6     HC    == HC_ini /\ [] [HC_nxt]_<<hr>>
line 7    -----------------------------------------------------------
line 8      THEOREM HC => []HC_ini
line 9    ========================================
```

line 0: Module name. In TLA codes are written as well-formed (structured) program pieces. Module hierarchy enables us to re-use them as required.
line 2: Natural numbers are defined for the module. Naturals.tla includes the definition of natural numbers.
line 3: Variable declaration. Variables do not have strict types. Variable types are defined after the first usage.
line 4: Initial condition. hr value is generated from this range for all starting states.
line 5: Next state action item is defined. This item is used to generate the next states.
line 6: Satisfaction rule. This rule means that $HC_{ini}$ and (for all future states) $HC_{nxt}$ must be satisfied together. For instance hr value cannot take a value out of the defined range. If this occurs, then checker stops and displays an error message.

**Figure 4. 11** TLA code example

## *4.5 Solution Verification*

As described before, one-to-one I/O constraint forces each participant to receive and send a single video signal at any given time for symmetric and low bandwidth conference sessions. This problem, namely multi-sourced multi-point multicast with one-to-one I/O constraint, is a well-known NP-hard problem in degree-constraint spanning trees and called Hamiltonian Path problem [37]. By developing NeViC, I propose an empirical evidence for the solution' s validity. For a formal verification of my approach, I used TLA (Temporal Logic of Actions) to specify the whole system and checked it with TLC (Temporal Logic Checker)[6]. I chose TLA and TLC because; they facilitate easy simulation and search for reachable states. Besides, both TLA and TLC have been developed especially for describing and checking small scale distributed models.

[6] See http://research.microsoft.com/users/lamport/

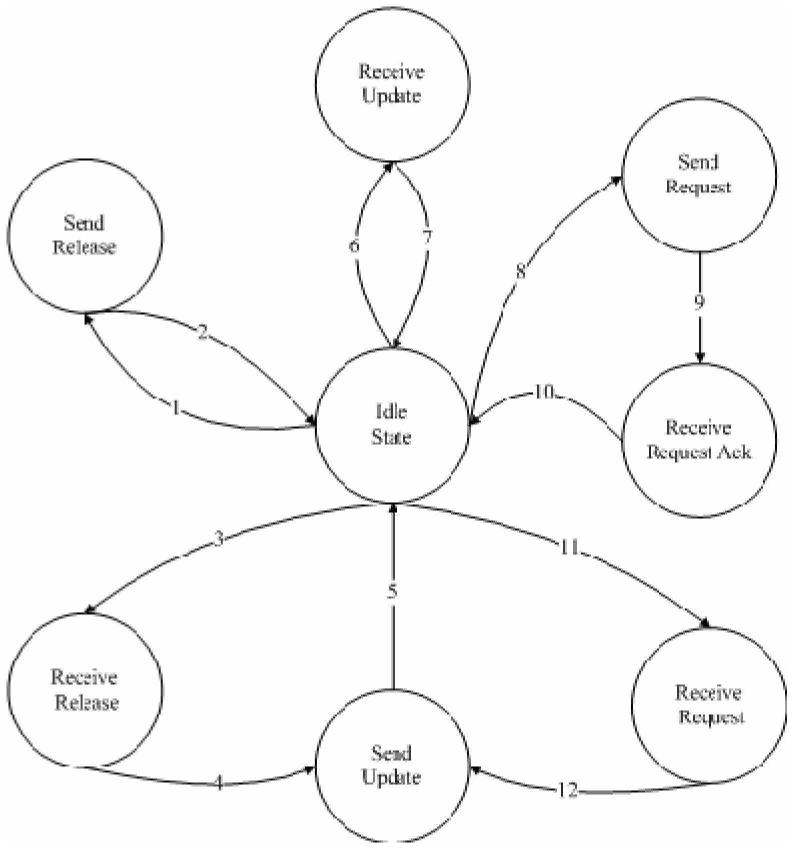
The temporal logic of actions (TLA) is logic for specifying and reasoning about concurrent systems. Systems and their properties are represented in the same logic, so the assertion that a system meets its specification and the assertion that one system implements another are both expressed by logical implication. TLA is very simple; its syntax and complete formal semantics are summarized in about a page at http://research.microsoft.com/users/lamport/ web site. Figure 4.11 displays a simple example for a simple clock. The formal TLA specification written in TLC (model checker for specifications written in TLA) is examined line by line.

It is important how TLC does the verification. TLC targets to generate a solution space where all possible states (generated from different combinations of the local variables) are present. Therefore, TLC uses a tree structure to hold each state generated at different steps. Tree is gradually generated and as soon as new states are created, they are controlled for the restrictions given at the TLA specification. If TLC catches a faulty state where no preventions are taken then TLC displays the error message and stops generating new states and quits. Since solution space can be wasted easily as the number of variables increases, it is too important to eliminate bulk programming blocks.

Figure 4.12 gives the summary for the verification technique used. For any state in the solution space, a (TLA defined) user can be in one of the eight states drawn with circles. All of the states are in parallel with their given aliases except the “idle” state. Here, idle state has no relation with the NeViC implemented idle state. In TLA, idle state means user state may change freely from one state to another. Therefore, according to the TLA declaration, a user can be in idle state while it is actually receiving video. Idle state accepts user to start a new transactions like demanding for video, applying for release etc. Thus, only the activable states (Directly reachable states from idle state) which trigger a new transaction can be reached from the idle state. For instance, as in figure 4.12, there is no way to reach send update state directly from idle state.

On the other hand, just to build a relationship between the verification model and Nebula actual implementation, figure 4.13 has been drawn which shows the state transitions for Nebula application. After a detailed examination, it is seen that figure 4.12 is a simple model generated from the original model shown in figure 4.13. The red lines in figure 4.13 shows the transitions supported in the verification model, too. Black lines are not considered while writing the TLA code. By using some central data types, transitions shown with black lines have been removed from the verification model. For instance, think about the “Receive Video Request Message” state model. Transitions 5 and 8 are not directly supported but by using central data they have been removed from the model.

Verification code given in the thesis appendix (see the disc), searches for some cases; (i) check if variables have out of range values, (ii) check if variables have undefined (e.g. null or alpha-numeric characters) values, and (iii) check if there is a deadlock situation for the whole system. Simulation has been run for several times and results showed that none of the problems titled above has been raised. To have a better understanding, final chapter gives the model checker results for number of total states and their execution times (for n users).



| ***From*** | ***To*** | ***Case*** | ***Action*** | ***Post Case*** | ***Related Links*** |
|---|---|---|---|---|---|
| Idle State | Send Release | Currently receiving video from remote user. | Send release message to the remote user. | Video transmission is stopped. Receiver is set as empty. | 1, 2 |
| Idle State | Receive Release | Currently sending video to remote users | Remove the remote user from local chain and send a reply back to the receiver. Also updates are notified. | User is removed the chain. If other users are available, video transmission goes on. Or else stopped. | 3, 4, 5 |
| Idle State | Receive Update | Currently member of a chain and an update message arrives. | Update the receiver information sent by the header user. | A new receiver value is set. | 6, 7 |
| Idle State | Send Request | Currently no video transmission is avilable for the demander. | Send a video request message to the remote user and waits for an ack. | If the request is granted then sender is updated. Otherwise everything remains same. | 8, 9, 10 |
| Idle State | Receive Request | Current, local user must not relay other user's video. | Make a decision if request is granted or not. Then send the ack to the demander. If request is satisfied then broadcast update message. | If request is granted then add user into the chain. Otherwise everything stays the same. | 11, 12 |

**Figure 4. 12** State transition graph for system verification

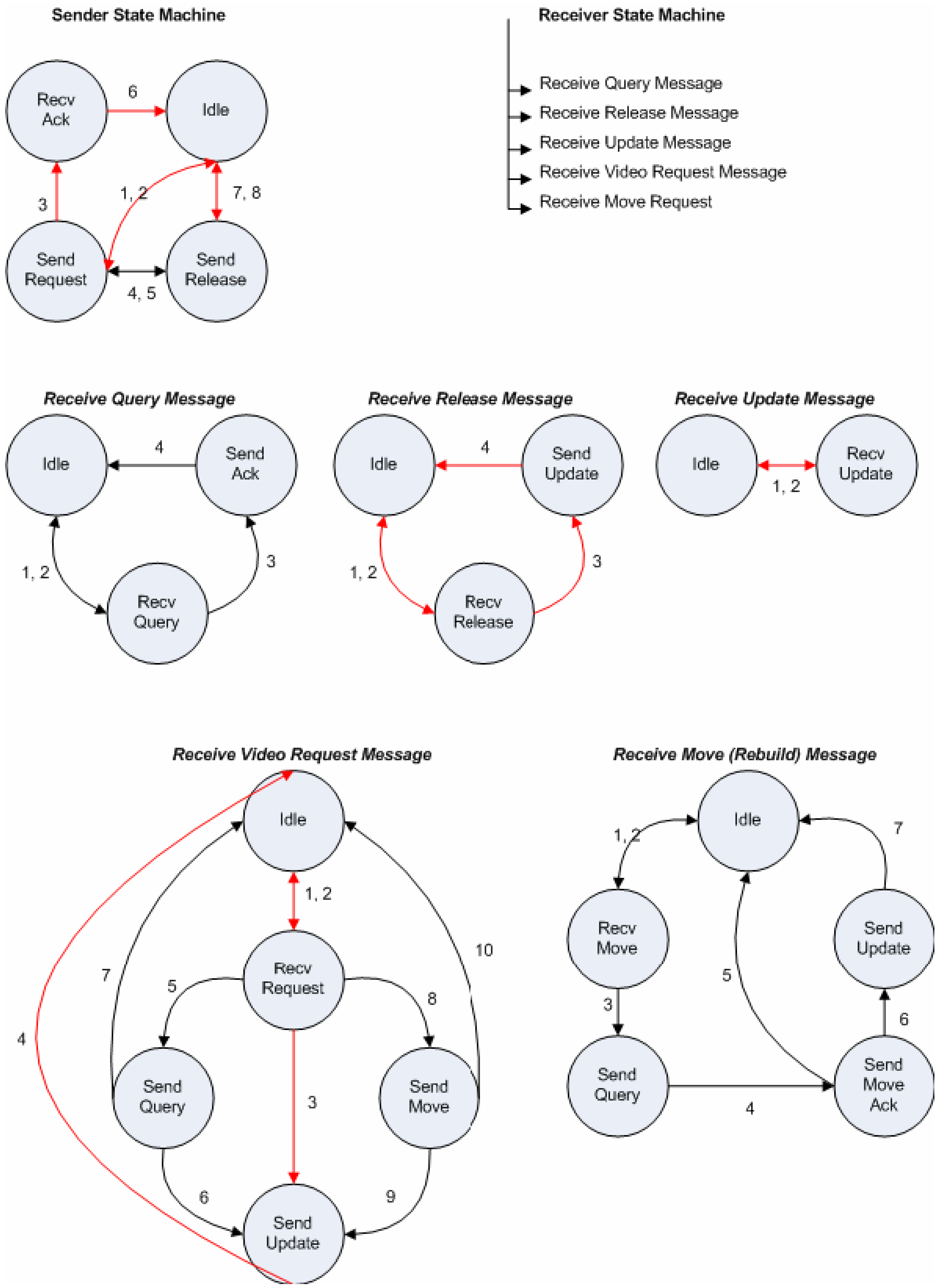


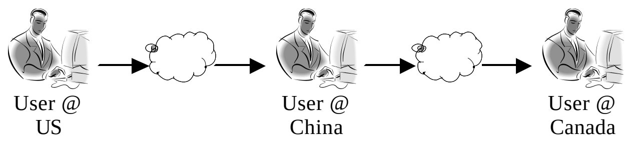
**Figure 4. 13** Nebula implementation state transition graph

# 5 OPTIMIZATION TECHNIQUES

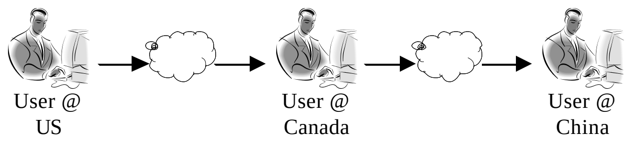
## 5.1 Why is optimization needed?

The key assumption in the previous chapter is that each peer can only receive and send a single video signal at a given time due to the bandwidth limitation. Thus, it treats each participant equally and chain construction process does not consider the participants' varying capabilities. However, the Internet is a heterogeneous environment where each host differs from the others with respect to its computing power, connection bandwidth and network location. As a result, ignoring these parameters while constructing video chains may lead to some problems e.g., increased delay, higher loss ratio and single point of failures. For instance, assume that three users with the same capacity located at different networks build a conference session. At a given time, user@US may have a chain as shown in figure 5.1.a. This is a problematic chain ordering because of the long reroute over China. Instead of using this chain, another chain given in figure 5.1.b, that takes into account network locations of participants, can decrease the total distance data travels and hence the overall delay. This reordering may also help to reduce the total packet loss rate because the longer routes will be more susceptible to congestion. For this purpose, an extension to the chain construction method presented in the previous chapter has been developed to search for the optimal chain orderings minimizing the total delay on a chain.

I ignore the computing power parameter of a participant and concentrate on the delay between each participant while re-constructing the video chains. Since I need transmission delays between peers to make a decision for the optimal video chains, I implemented a delay measurement protocol which is based on clock synchronization



.*a*) Erroneous ordering

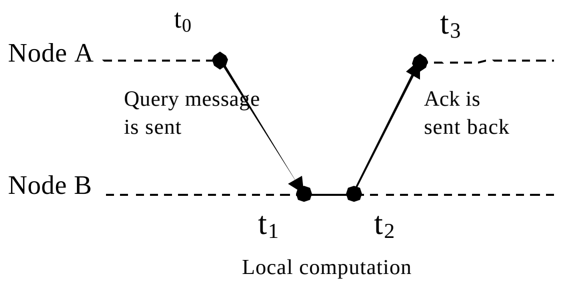


*b*) Reconstructed ordering

**Figure 5. 1** Chain reconstruction example

algorithm described in [38] and shown in figure 5.2. In the figure, node-A sends a query message to node-B and node-B replies this message with the time value used for local computation ($t_2 - t_1$). After node-A receives acknowledgement from node-B, it can calculate average network delay between nodes A and B using the formula given in the figure. In my prototype, each peer keeps a globally known delay matrix specifying the delays between each conference participant at a given time. The delays are measured for a participant as soon as it registers for the conference. All participants except the new registered one waits for a random time (to eliminate collisions) and run the delay calculation mechanism for the newly registered participant. All participant matrices are updated after the delay measurements are complete. This is needed because I chose a Two Phase Heuristic for chain reconstruction.

In order to find the optimal ordering of the peers on a long chain considering the overall delay, I have implemented the "Two Phase Heuristic" described in the next section. This heuristic can be used to determine degree constrained minimum spanning trees (MST) on an undirected graph where the degree constraint is defined as the maximum number of edges that can be connected to a node. At each step, a node which violates the degree constraint (e.g., more than one output edge in my application) is chosen and its degree is tried to be decreased by re-locating its output somewhere else on the chain. This calculation gives a chain whose total delay between the head and the last nodes is in



**Delay(A,B) =[ $t_3 - t_0 - (t_2 - t_1)$ ] / 2**

**Figure 5. 2** Delay measurement protocol

general smaller. A drawback for this algorithm is that it may not find an implementable solution when it tries to move the last node of the unordered chain to the middle. So as an additional task, the generated solution has to be checked to see if the last node of the unordered chain is usable in this new configuration. If not (because it is sending its own video to another node), then the optimization result can’ t be used. In such cases, full enumeration (O (n!)) may be tried to find the optimal ordering.

A separate thread is reserved for chain re-construction and it runs the shortest path algorithm mentioned above periodically. Whenever a participant joins into a chain, it is tried to be located immediately without regarding the optimized network delays. The re-construction for optimized delays can be done at a more suitable time where the chain head stands idle. This concept is called as soft-join.

For now, delays are assumed to be static and they are calculated only once at the conference login. The delays may change during a conference session. RTCP may be used to assist in handling such cases. So as a further improvement, the delay optimization process may be run again when the video jitter reported by RTCP changes dramatically.

## *5.2 Techniques for Optimization*

Is it fair to load each participant equally or what happens and what improvements will be gained if some participants can serve more than one end points? It is clear that this handles many deadlock cases in the previous model. Besides, new approach should cover the previous model so one-to-one I/O should be solved using the general model. In literature, finding a multicast tree for multi-sourced conference is known as minimum "cost degree-constrained spanning tree" problem and identifying such a tree is computationally NP-hard [39]. Thus, several heuristics have been offered to find a qualified solution under time or computing power constraints. Heuristics that I utilize for MP videoconferencing are described next.

### 5.2.1 Two Stage Heuristic

I consider a two-stage heuristic algorithm [40] for the solution of the asymmetric multipoint video conferencing problem. The first stage involves constructing a minimum spanning tree (MST) from a complete graph (this graph is constructed using global delay matrix described in the optimization part above) without considering the degree constraints. Then the second stage modifies the spanning tree to satisfy the given I/O constraints (degrees) for all participants. The details of the algorithm and its procedures are described in figure 5.3. In the original algorithm, the degree constraint is the same for all nodes. In my solution, nodes may have different constraints. In the first step, an MST can be calculated in polynomial time using greedy algorithms like Kruskal' s [41] or Prim' s [42]. After implementing the first step, generated tree is processed to adjust the I/O constraints on nodes. This step involves selecting a node that violates the I/O constraints and disconnecting the highest-cost edge of this node. Next, the subtree created by this operation is re-attached to the current tree. This operation goes on until all nodes have valid number of I/O connections. This heuristic is not guaranteed to find a solution under all cases. That is, when it fails to find a solution, this does not mean that no solutions are available. However, I will declare no solution in these situations.

```
Find an efficient MST using Kruskal algorithm
Call degree-resolve () method for calculated MST

void degree-resolve(){
        1- Select a participant (let p) from the calculated MST that does not
        satisfy the degree constraint. If two or more candidates exist then select
        one randomly
        2 – Remove an out-edge link (let l) which has the largest delay from p.
        This step divides our spanning tree into two sub trees
        3- Try to attach the second tree (which does not include the source) to
        the first one. While combining trees, consider low-delay and degree
        constraints
        4- Repeat this procedure until each participant satisfies the degree
        constraints defined for themselves.
}
```

**Figure 5. 3** Algorithm for two-stage heuristic

The advantages of this approach are that it is simple and easy to use. Besides, it puts not only degree constraints but also delay constraints on the spanning trees. Results of this algorithm show that it is feasible to use when the participant count is equal to or less than 20.

### 5.2.2 Genetic Algorithm (GA) Based Heuristic

This part offers an evolutionary approach for finding efficient degree-constraint MSTs defined in [43, 44]. As in the previous approach, original form of this algorithm assumes that the degree constraints are homogenous for all participants. I modify the algorithm assigning different constraints on each participant. Since this solution is based on GA, defining the GA representation and setting GA parameters like population initialization, elitism, end criteria etc. have profound impact on the solution quality and interval.

```
void initialize() {
        T = {}
        For all edges (i,j) member in E with random order do
                If deg(i) < $d_i$ and deg(j) < $d_j$ and not conected(i,j,T)
                        T = T union {(i,j)}
                        If |T| == |E|-1
                                return
}
```

**Figure 5. 4** Initialization algorithm for degree-constraint ST

Thinking STs as undirected graphs allows us to see whole picture easily. From now on I call STs as chromosomes, and I assume that a well-formed GA representation technique is used to define STs. The algorithm needs an initialization step for generating whole population. Initial chromosomes are created using a modified version of Kruskal' s algorithm demonstrated in figure 5.4. This algorithm [43] allows us to construct a feasible ST where the degree constraints are ensured. Here T is initially an empty graph and E is the complete graph based on a global delay matrix (see optimization part). i and j are the vertices that are controlled and deg(i) and deg(j) is the vertex' s current load. $d_i$ and $d_j$ are the constraints defined for the vertices. The connected () operator checks if there is any path between the selected peers. An important criterion for this step is the sort function implementation. The edges are sorted according to a cost function shown in Eq. 1. This cost function combines edge delay and degrees of its vertices. In the cost function, *C* is the cost between vertices i and j while *Wi* is the degree constraint available for the vertex at one end of the edge under consideration and $\mu$ is a constant (0 or 1) which is used to adjust the relative importance of the cost and the degree. If $\mu = 0$ then the ST function tries to utilize all edges. This leads to a case where most of the outputs are used in the ST. On the other hand, if $\mu = 1$ then the function tries to find a solution where available outputs are minimally engaged so that they can be kept for future usage.

$$Ci,j' = Ci,j + ( Wi + Wj ) \times \mu \qquad \textbf{Eq. 1}$$

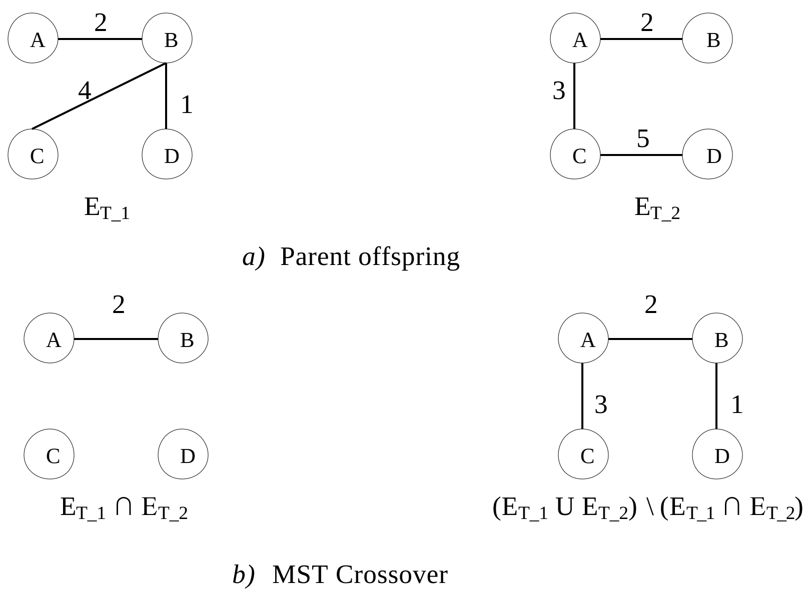


**Figure 5. 5** A simple crossover example. Each node has a constraint of 1 input and 2 outputs

A major step of the GA based approaches is the application of a crossover operator which enables shared representation of different offspring in a population. In my approach, the development of the crossover operator was substantially guided by the idea to produce a new degree constraint spanning tree by inheriting as many edges as possible from two parental trees. Details of the algorithm and a pseudo code are present in [43] for the crossover operator.

In the first step, ET (set of edges for next offspring) is initialized with the shared edges between parental trees (chromosomes) $E_{T\_1}$ and $E_{T\_2}$. These shared edges will be present for the next offspring. In the second step, all edges contained in both parents $E_{T\_1}$ and $E_{T\_2}$ (but not in both) are checked for inclusion. As in the initialization process, edges are sorted using cost function and tried sequentially if a valid spanning tree is constructed or not. A further step is needed since a valid spanning tree may not be constructed. Some edges might be incomplete after merging $E_{T\_1}$ and $E_{T\_2}$. Hence, as a final step, all edges from the complete graph (not including $E_{T\_1}$ union $E_{T\_2}$) are sorted and tried sequentially to construct a valid ST. This final step guarantees us to find a valid ST.

The final operator of the GA based approaches is mutation and in our solution edge mutation is based on the principle of adding a new edge and deleting the current one which makes the ST feasible. The mutation operator suggests selecting two vertices that are not directly connected. Before connecting vertices, a path between them has to be found and an edge on this path is randomly selected and removed.

Figure 5.5 shows a simple example for the crossover operation. Parental offspring are displayed in figure 5.5.a and figure 5.5.b implements the crossover operation using the rules described above.

An important optimization for this approach is to initialize population with similar STs. This guarantees a fast search to find a steady state where chromosomes stay the same (offspring have more shared edges so constructing new valid STs may take less time). So crossover and mutation operators do not degrade the performance.

Experiments show that this solution can be applied to any conference without considering the participant count. Simulation results are much better than the two stage heuristic especially according to time [43]. However, implementation of this approach is more complicated because of adjusting the GA parameters. Therefore two-stage heuristic has been selected as the main strategy for chain reconstruction in this thesis.

# 6 CONCLUSIONS

I implemented, verified and tested an optimal system for peer-to-peer multipoint video conferencing. The system can be used to extend almost any point-to-point video conferencing system to a multipoint one with minimal increase in the complexity and no additional hardware support. The system architecture allows using off-the-shelf components for most of the system elements. I tried to stay compliant with the standards whenever possible, made additions if required. All codes written for the experiments mentioned here are located under the appendix part (see the disc).

## *6.1 Solution Availability Test*

This test considers the number of deadlocks that might occur for a conference session in theory and practice. Simulation results (practical values represented using red line) obtained so far on the percentage of unachievable configurations as a function of the number of conference participants are depicted in figure 6.1. When compared to the theoretical values (represented using blue line), simulation results show that, as the number of participants gets larger, the percentage of unachievable configurations stays constant, or increases very slowly. This is mainly due to the fact that in a dynamic conference setting with requests concurrently triggered by all participants, all possibilities of chain configurations are not equally likely to occur since simulation characteristic is related with random event generation. Defined events for the simulation are video request and release. As soon as the conference starts, all participants' states are set to idle. For a predetermined time interval, each participant generates an event periodically. Thus a participant' s chain usually is small. Since for each time interval (mostly for a participant) one or two requests arrive while its current receiver releases its resource in the same interval. As a consequence, chains are not too long to lead to a deadlock situation. Naturally, as the number of participants increase, for each time interval, the collision possibility (requesting to the same participant for video transmission) decreases. The computational burden caused by the peer-to-peer application on the system is observed to be insignificant (less than 1% CPU time on a 2.4 GHz Pentium).

On the other hand, a theoretical work has been done just to see how the system reacts for single source requests. For this reason, I tried to give a mathematical formula for chain construction concentrated on unique chains. Since all nodes are symmetric, it is not true to enumerate all cases as told before. I mean when we enumerate all cases actually I repeated most of the cases. Therefore, here I totally concentrated on a single source and its possible states. Figure 6.4 shows a way to generate unique chains for a single source without repeating the same orders. And also the mathematical calculations can be found in figure 6.4. To map the results with the ones shown in figure 6.1, assume that we have 8 users in the chain. According to the formula our values are 16/1 (requestor' s output is idle) and 9/8 (requestor' s output is busy) where the first values are valid counters and the successors are invalid counters. That means invalidity equals to 9/34 which is nearly %27 and mapped with the simulation results drawn with red line in figure 6.1.

It is a little interesting that numbers of states are not matched for the theoretical values. For instance, according to the table, 3 users case generates 27 distinct states. However, there are 54 controlled states for deadlock. Vagueness is originated from the different orderings in the chains. To make it clear, it has to be known how the simulator handles the abstract chains. Simulator keeps two arrays; (i) first array keeps the IDs of the participants (receivers) and, (ii) the second one keeps the senders' IDs. These two arrays are one-to-one related. Assume that arrays are; <A, B, C> and <B, C, 0>. According to this example, user-A receives video from user-B. On the other hand, C is not receiving video and this is represented using ' 0' character. After generating a state as shown above, simulator tries to check if the given state is valid or not. Control criterion is one-to-one I/O restriction. If the state is invalid then the operation is terminated. Otherwise, simulator tries to enumerate all sub-cases (pivot element, selected at the simulation startup, requests video from all other users sequentially) and traces each sub-case result. This flow causes difference between generated states and checked states.

## 6.2 Solution Performance Test

NeViC performance is measured and logged for the highly-loaded case. Figure 6.2.a shows the run-time cost of the peer-to-peer prototype application for 15 seconds while both incoming and outgoing channels are busy. CPU idle time (brown line) has been measured and it shows that our peer-to-peer application leads to a 5-7% computational load. Another logged criterion is network utilization. Just for the sake of dial-up users, the peer-to-peer prototype encodes 3 frames which are encoded as MPEG4 using ffmpeg codec (freely distributed @ www.sourceforge.net). In this case, total UDP transmission load is almost 60-70 Kbps. These values are quite good for any ordinary users who want to have a video conference session using modem over the Internet. Strictly speaking, most of the current video application does not obey these constraints.

As another measurement, a conference session with three users was created and all users were sorted on a chain. In other words, one user is selected as the video sender and the others are assigned as the video receivers. Figure 6.2.b displays the test machine configurations and related simulation results for each user. At first glance to the simulation results, someone strongly possible asks that “Why does not the load on the relay node bigger than the others. It has two responsibilities where the others have only one”. However response is kept in the nature of NeViC application. Ignoring the status of the user, NeViC always do the same operations even if the user sits in idle state. That means NeViC has an operation chain like, (i) sample video, (ii) send it to the RTP gateway, (iii) receive video from RTP gateway and display it. Assume that user is in idle state. In this case, user samples its own video; send it to the RTP gateway and RTP gateway will return it back to the display engine. Therefore all nodes have approximately the same cost.

## 6.3 Solution Verification Test

Figure 6.3.c shows the verification results of TLA/TLA+ specification of the peer-to-peer model. TLC model checker tested the prototype application for four criteria; (i) deadlock situations, (ii) silly expression assignments, (iii) unwanted value assignments, and (iv) states' liveness conditions. One-to-one I/O prototype has been successfully verified since no problem is detected by the TLC model checker. Figure 6.3.c shows the experiment results run for 3, 4, 5, 6 users. Total numbers of the generated and unique states are given with the simulation results. As it can be seen from the statistics, as the number of users increase (in simulation), there is a burst in the (generated) state number. Therefore, I cannot verify the cases that include more than six users. Theoretically, four-user case will be enough to traverse the whole tree shown in figure 4.2 (means all possible decision paths are used). Therefore if four-user case is satisfiable, then it shall be a good sign but not enough to test all possible cases if there are more users available. Finally I need to talk about the 3rd control criteria, unwanted value assignments. Verification code is capable of catching erroneous assignments like null or alpha-numeric assignments in user chains. But unfortunately it is not possible to catch unwanted numeric assignments in the chains like having an undefined node ID or duplicated node IDs in the chains because of the control criterion. For the sake of simplicity it was defined as "allowing numeric values in the chain without looking if they are valid or not".

## 6.4 Optimization Test

I verified the effectiveness[7] of the optimization algorithm that is used for reducing the delay on long chains (Two Stage Heuristic) by performing tests on randomly generated

[7] Effectiveness (feasibility) in this context, is the ratio of the "cost gained / original cost"

chains and the results are displayed in figures 6.3.a and 6.3.b. For each test (that is for each entry of the horizontal axis), 5000 sample chains have been generated and all of the algorithms have been forced to find a solution. Horizontal axis shows the participant count and the vertical axis shows the effectiveness percentage. Figure 6.3.a shows the comparison of the three algorithms offered for tree construction. First algorithm Kruskal MST (pink line) has the best results since it does not take into account the degree constraints. On the other hand, for a chain, the best solution can be found by enumerating (blue line) all states and selecting the minimum cost state. Finally Two Phase Heuristic (yellow line) has an effectiveness level nearly approaching 50% for the original cost.

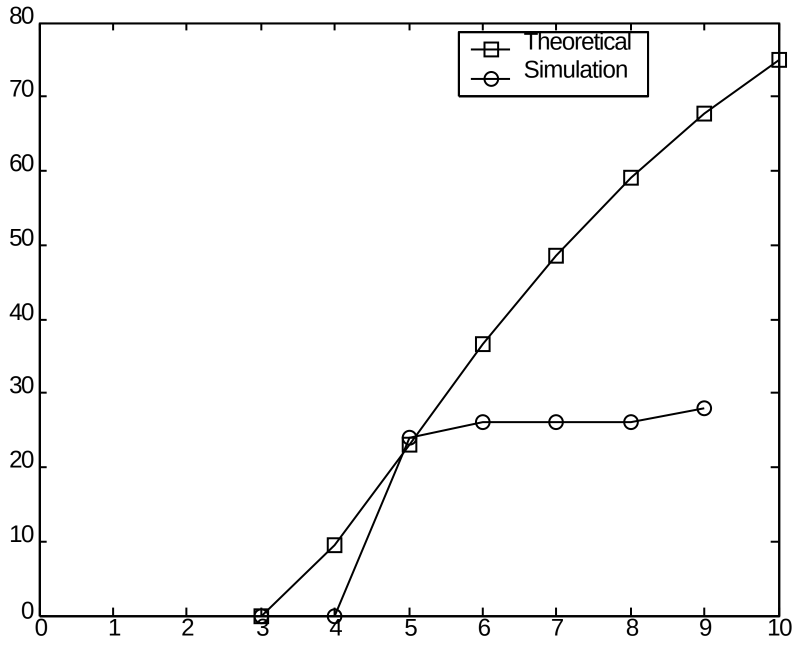
On the other hand, Figure 6.3.b shows the CPU times for the enumeration and the heuristic solutions. As the number of participants' increases, enumeration suffers from the time needed to find the best solution. Unlike enumeration, heuristic based search for chain reconstruction guarantees to find a effective solution in a reasonable time.

All experiments for the optimization problem are run on a test machine whose configuration is summarized as "Intel (R) Pentium (R) 4 CPU 2.60 GHz, 512MB of RAM, MS Windows XP Pro v.2002". Besides, appendix (see the disc) includes the program codes written for the mentioned experiments.

## *6.5 Future Work*

This thesis offers a new way for implementing multipoint video conferencing. Offered approach has also been tested and verified using empirical and formal techniques. For the empirical evidence, protocols offered have been developed under NeViC application. On the other hand, formal verification has been done using TLA/TLC. However, there is still some gray-area that is consciously not fulfilled. Those problems should be thought as future work and can be listed as;

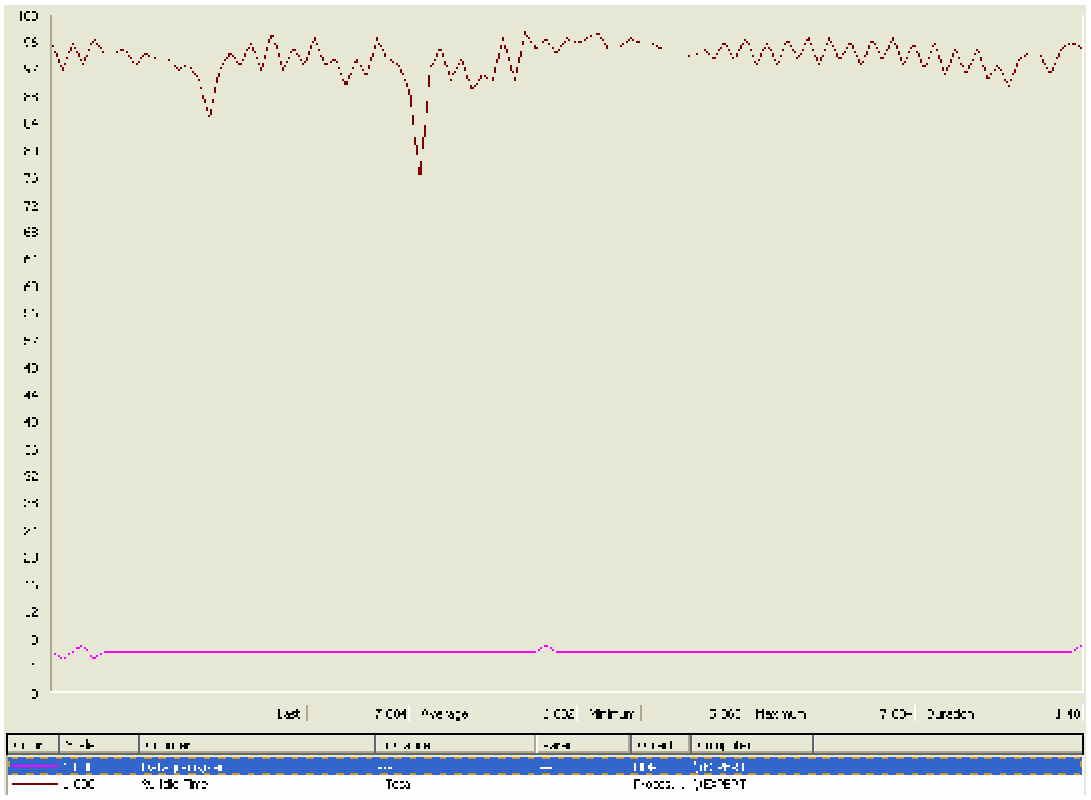
- Current application supports one-to-one multipoint video conferencing. Although it offers a solution (solution quality is tested via simulations) for one-to-many multipoint video conferencing, it has not been added into the current application. For a future work, this option might be enabled in the application.
- Current application uses RTP messages for video transmission. However, RTCP messages are disabled and delays between end-points are assumed as static (never changed over time). As a future work, RTCP messages should be enabled and delays should be measured using them (dynamic delays).
- As mentioned in the first chapter, some issues like security, platform independency etc. are ignored since each of these topics can be a separate research. Therefore, future work should address these topics.



| 🟦 | Σ(All ***States)*** | Σ(Valid ***States)*** | Σ(Invalid ***States)*** | Σ(Deadlock ***States)*** | Σ(Deadlock-***free States)*** |
|---|---|---|---|---|---|
| ***3 Users*** | 27 | 27 | 0 | 0 | 54 |
| ***4 Users*** | 256 | 232 | 24 | 66 | 630 |
| ***5 Users*** | 3125 | 2405 | 720 | 1692 | 7928 |
| ***6 Users*** | 46656 | 29616 | 17040 | 34840 | 113240 |
| ***7 Users*** | 823543 | 423283 | 400260 | 712110 | 1827588 |

| 🟥 | Σ(All ***Requests)*** | Σ(Granted ***Requests)*** | Σ(Rejected ***Requests)*** | Σ(Others) |
|---|---|---|---|---|
| ***4 Users*** | 788 | 780 | 5 | 3 |
| ***5 Users*** | 809 | 597 | 196 | 16 |
| ***6 Users*** | 1001 | 725 | 264 | 12 |
| ***7 Users*** | 1210 | 789 | 283 | 138 |
| ***8 Users*** | 1176 | 859 | 304 | 13 |
| ***9 Users*** | 1600 | 1139 | 438 | 23 |

**Figure 6. 1** Theoretical and simulation results for unachievable configurations' percentage (y) of all configurations for a given number of participants (x)

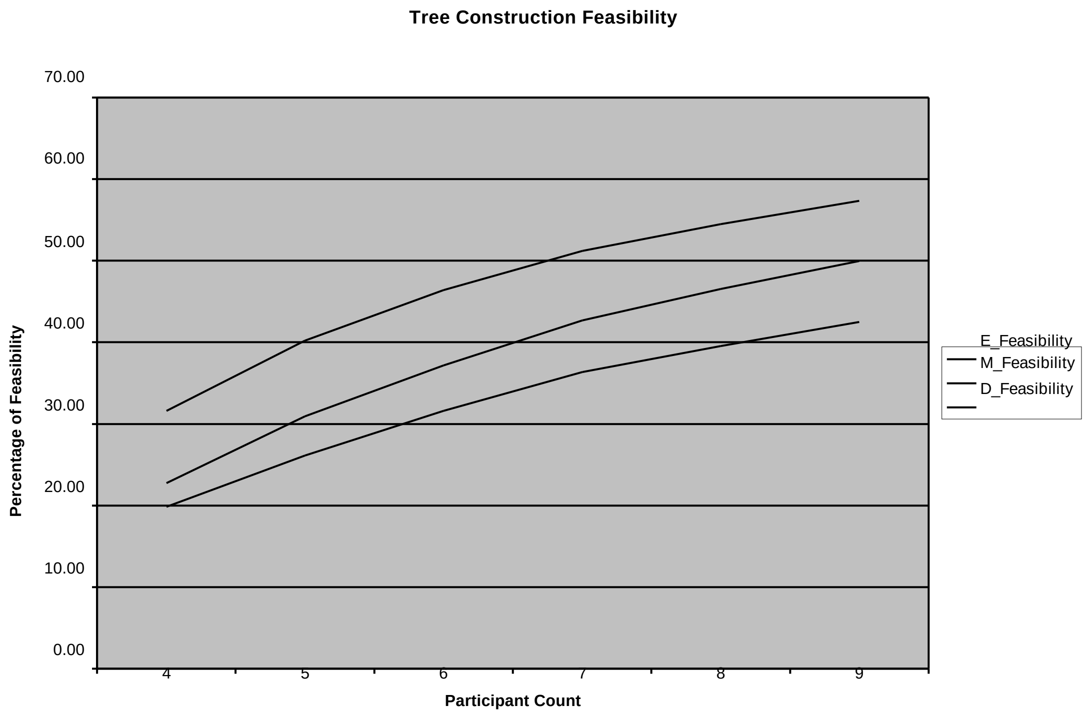


.*a*) System log for runtime of the codec. Controlled parameters are CPU idle time (brown-scale 1.0) and UDP bytes\sec (pink-scale 1.0).
<u>Test machine is Intel (R) Pentium (R) 4 CPU 2.60 GHz, 512MB of RAM, MS Windows XP Pro v.2002</u>

| ***Machine ID*** | ***Machine Configuration*** | ***Avg. CPU Load*** | ***Avg. Memory Load*** | ***State*** |
|---|---|---|---|---|
| ***1*** | P-IV 2.40 GHz + 2.00 GB of RAM | 8% | 15 MB | Send |
| ***2*** | P-IV 2.40 GHz + 512 MB of RAM | 4% | 16 MB | Receive&Relay |
| ***3*** | P-IV 2.40 GHz + 248 MB of RAM | 7% | 17 MB | Receive |

.*b*) Performance benchmarks on different platforms.

**Figure 6. 2** NeViC performance results



*a*) Feasibility benchmark of 3 methods; Kruskal MST, Enumeration of all chains, Two Phase Heuristic.

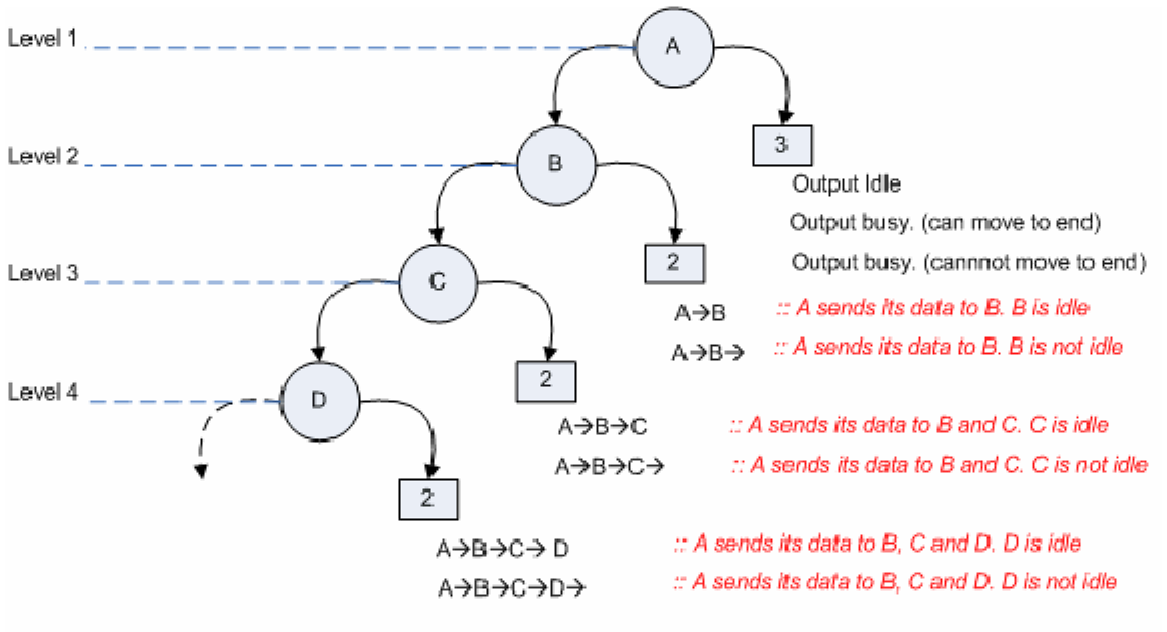
| ***(in msecs)*** | ***ETREE*** | ***DTREE*** |
|---|---|---|
| **4** | 1.25 | 1.42 |
| **5** | 1.39 | 0.48 |
| **6** | 4.39 | 0.63 |
| **7** | 31.87 | 0.62 |
| **8** | 207.8 | 0.31 |
| **9** | 1621.55 | 1.1 |

.*b*) CPU times in milliseconds for algorithm run times. Benchmarked algorithms are (E)numeration and (D)egree Constrained using Two Phase Heuristic

| ***Node #*** | ***Reached State*** | ***Unique State*** | ***Search Depth*** | ***Search Result*** |
|---|---|---|---|---|
| ***3*** | 3337 | 1438 | 37 | Satisfied |
| ***4*** | 102189 | 36117 | 50 | Satisfied |
| ***5*** | 3566161 | 1026216 | 83 | Satisfied |
| ***6*** | 140972179 | 33379651 | 102 | Satisfied |

*c*) Verification results for one-to-one I/O model TLA specification

**Figure 6. 3** Chain feasibilities (effectiveness), CPU costs and verification results


Level 1
Level 2
Level 3
Level 4
A
B
C
D
3
2
2
2
Output Idle
Output busy. (can move to end)
Output busy. (cannot move to end)
A→B
:: A sends its data to B. B is idle
A→B→
:: A sends its data to B. B is not idle
A→B→C
:: A sends its data to B and C. C is idle
A→B→C→
:: A sends its data to B and C. C is not idle
A→B→C→ D
:: A sends its data to B, C and D. D is idle
A→B→C→D→
:: A sends its data to B, C and D. D is not idle

| | X's Output is idle | | | X's output is busy | |
|---|---|---|---|---|---|
| | Valid | Invalid | | Valid | Invalid |
| Level 1 | 2 | 1 | | 2 | **1** |
| Level 2 | 2 | 0 | | 1 | **1** |
| Level 3 | 2 | 0 | | 1 | **1** |
| Level 4 | 2 | 0 | | 1 | **1** |
| ... ... ... ... ... | | | | | |
| ... ... ... ... ... | | | | | |
| Level n | 2n | 1 | | n+1 | n |

X is the requestor Node
This table examines possible cases when X's output is idle or busy.

**Figure 6. 4** Formulizing Nebula Chains

# BIBLIOGRAPHY

[1] ITU-T Study Group XV, "Recommendation H.231, Multipoint Control Units for audiovisual systems using digital channels up to 1920 kbit/s", March 1993.

[2] M. R. Civanlar, R. D. Gaglianello, G. L. Cash, "Efficient Multi-Resolution,Multi-Stream Video Systems Using Standard Codecs", Journal of VLSI Signal Processing, 1997.

[3] M.R. Civanlar, O. Ozkasap, T. Celebi, "Peer-to-Peer Multipoint Video Conferencing", ICIP, October 2004.

[4] T. Celebi, M.R. Civanlar, O. Ozkasap, "P2P Multi Point VideoConferencing in the Internet" , ,PDCN, February 2005

[5] Yang-hua Chu, Sanjay G. Rao, Srinivasan Seshan and Hui Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture", Proceedings of ACM SIGCOMM' 01, San Diego, California, August 2001.

[6] M. Hosseini, N. D. Georganas, "Design of a Multi-sender 3D Videoconferencing Application over an End System Multicast Protocol" , ACM-Multimedia 2003, Berkeley, CA, November 2003.

[7] D. Cohen. "Specifications for the Network Voice Protocol", Network Working Group, RFC 741, November 1977.

[8] See http://www.columbia.edu/~rh120/ch106.x08

[9] J. Forgie, " ST - A Proposed Internet Stream Protocol" , IEN-119, M.I.T. Lincoln Laboratory, September 1979.

[10] M. Handley, C. Perkins and E. Whelan, " Session Announcement Protocol", Network Working Group, RFC 2974, October 2000.

[11] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, " Session Initiation Protocol", Network Working Group, RFC 2543, March 1999.

[12] M. Handley and V. Jacobson, "Session Description Protocol, Network Working Group", RFC 3261, April 1998.

[13] ITU-T Study Group XV, “Recommendation H.261: Video codec for audiovisual services at px64 kbit/s” , tech. rep., ITU-T, Geneva, 1993.

[14] R. Roy, AT&T, "H.323 Mobility Architecture and Protocol for Terminal, User and Service Mobility," ITU Document APC-1652, Oct. 1999.

[15] S. McCanne, V. Jacobson, "VIC: A Flexible Framework for Packet Video," in Proceedings of ACM Multimedia, 1995.

[16] See http://csperkins.org/rat/index.html.

[17] H. Schulzrinne, A. Rao, R. Lanphier, "Real Time Streaming Protocol (RTSP)", Network Working Group, RFC 2326, April 1998.

[18] See http://www.napster.com

[19] See http://www.limewire.com/english/content/home.shtml

[20] S. Deering and D. Cheriton, “Multicast Routing in Datagram Internetworks and Extended LANs. In ACM Transactions on Computer Systems”, May 1990.

[21] Y.H. Chu, S. G. Rao, and H. Zhang, “A Case for End System Multicast. In Proceedings of ACM SIGMETRICS”, June 2000.

[22] P. Francis. Yoid: “Extending the Multicast Internet Architecture”, White paper at http://www.aciri.org/yoid/, 1999.

[23] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrastructure for real-time applications," in Proc. IEEE Infocom, June 2003.

[24] Suman Banerjee, Bobby Bhattacharjee, Christopher Kommareddy, “Scalable Application Layer Multicast", Proceedings of ACM SIGCOMM’ 02,Pittsburgh, Pennsylvania, August 2002.

[25] S. Shi and J. Turner, “ Routing in overlay multicast networks,” in Proceedings of Infocom, June 2002.

[26] Daniel Brookshier, Darren Govoni Navaneeth Krishnan, “JXTA: Java P2P Programming” , pp.8-60, Indiana, 2002.

[27] Y. Chu, S. Rao, and H. Zhang, “A Case for End System Multicast”, In Proceedings of ACM Sigmetrics, June 2000.

[28] S.Floyd, M. Handley, J. Padhye, and J. Widmer, " Equation based congestion control for unicast applications" , in Proc. ACM SIGCOMM, August 2000.

[29] M. Hosseini, N. D. Georganas, "Design of a Multi-sender 3D Videoconferencing Application over an End System Multicast Protocol" , ACM-Multimedia 2003, Berkeley, CA, November 2003.

[30] Duc A. Tran, Kien A. Hua, Tai Do, ZIGZAG, "An Efficient Peer-to-Peer Scheme for Media Streaming", Proceedings of IEEE Infocom 2003.

[31] S. Sheu, Kien A. Hua, and W. Tavanapong, "Chaining: A generalized batching technique for video-on-demand", Proc. of the IEEE Int' l Conf. On Multimedia Computing and System, pp. 110–117, Canada, June 1997.

[32] Venkata N. Padmanabhan, Helen J. Wang, Philip A. Chou Kunwadee Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking", Proceedings of NOSSDAV' 02, Miami, Florida, May 2002.

[33] Mark Handley, Jon Crowcroft, Carsten Bormann and Jörg Ott, "Very large conferences on the Internet: the Internet multimedia conferencing architecture", The International Journal of Computer and Telecommunications Networking, Vol 31, No 3, pp 191-204. Elsevier, North Holland, 1999.

[34] See http://ffmpeg.sourceforge.net/index.php

[35] See http://m4if.org/

[36] J. Chesterfield, E. Schooler, "RTCP Extensions for Single Source Multicast Sessions with Unicast Feedback", Internet Draft, draft-ietf-avt-rtcpssm-03, March 2003.

[37] F. Rubin, "A Search Procedure for Hamilton Paths and Circuits", Journal of the ACM (JACM), vol. 21 , Issue 4, pp: 576 - 580, 1974.

[38] A.S. Tanenbaum, M.V. Steen, "Distributed Systems – Principles and Paradigms", pp. 247-24, New Jersey, 2002.

[39] R. Ravi, M.V. Marathe, S.S. Ravi, D.J. Rosenkrantz and H.B. Hunt," Many birds with one stone: Multi-objective approximation algorithms", in Proc. 25th Annu. ACM STOCS, pp.438-447, 1993.

[40] Nobuo Funabiki, Jun Kawashima, Akihito Kubo, Kiyohiko Okayama,and Teruo Higashino, “A proposal of a two-stage heuristic algorithm for peer-to-peer multicast routing problems in multihome networks” , The Fifth Metaheuristics International Conference (MIC-2003), pp. 20- 1-6, August 2003.

[41] J.B. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem”, Proceedings of the American Mathematics Society, vol. 7(1), pp. 48-50, 1996.

[42] R. Prim, “Shortest connection networks and some generalizations”, Bell System Technical Journal, vol. 36, pp. 1389-1401, 1957.

[43] G. R. Raidl, “An efficient evolutionary algorithm for the degreeconstrained minimum spanning tree problem”, Proceedings of the 2000 IEEE Congress on Evolutionary Computation, pages 104-111, IEEE Press, 2000.

[44] G. R. Raidl and B. A. Julstrom, “A weighted coding in a genetic algorithm for the degree- constrained minimum spanning tree problem”, Proceedings of the 2000 ACM Symposium on Applied Computing, pages 440-445, ACM Press, 2000.

[45] J. Brooks, M. Jurczk, “ Creating Edge-to-Edge Multicast Overlay Trees for Real-time Video Distribution” , International Conference on Internet Computing, 371-377, 2003.