Scalable Service Oriented Architecture

For Audio/Video Conferencing

by

AHMET UYAR

B.S., Cukurova University, Turkey, 1996

M.S., Syracuse University, USA, 1999

DISSERTATION

Submitted in partial fulfillment of requirements for the degree
of Doctor of Philosophy in Computer Science
in the Graduate School of Syracuse University

May 2005

Approved	Professor Kishan G. Mehrotra	Professor Geoffrey C. Fox
Date		

Abstract

Audio/Video conferencing over Internet is increasing rapidly with the increase on the available network bandwidth and computing power. Even the cell phones will have the capacity to participate in videoconferencing sessions in the near future. This requires scalable and universally accessible videoconferencing systems. Increasingly diverse set of endpoints with various capabilities from cell phones to room based systems should be supported in such videoconferencing sessions. However, developing videoconferencing systems over Internet is a challenging task, since audio and video communications require high bandwidth and low latency. In addition, the processing of audio and video streams is computing intensive. Therefore, it is particularly difficult to develop scalable systems that support high number of users with various capabilities. Current videoconferencing systems can not fully address the problem of scalability and universal accessibility. Therefore, in this thesis we investigate scalable service oriented architecture for audio/video conferencing.

We propose using a publish/subscribe event broker network for the distribution of real-time audio/video streams in videoconferencing sessions and investigate the issues pertaining to scalability, performance, data representation and routing. We identify the requirements for the delivery of real-time audio/video traffic and incorporate necessary changes to the event delivery middleware. We present the results of extensive performance tests for the delivery of audio/video streams for both large and small size meetings in single and multiple broker settings. In addition, we propose a scalable and flexible architecture for meeting management and media processing, that can grow dynamically to provide services to higher number of users.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my gratitude to my advisor, Professor Geoffrey C. Fox, for his guidance throughout my research and for his wise and acute observations on how to improve my work. I want to express my genuine thanks to Dr. Shideep Pallickara and Dr. Wenjun Wu for their advising and help throughout my research. I also want to thank Dr. Gordon Erlebacher for arranging the computing facilities at FSU to perform my tests.

I gratefully acknowledge Dr. Geoffrey C. Fox, Dr. Kishan Mehrotra, Dr. Ehat Ercanli, Dr. Roman Markowski, Dr. Shrideep Pallickara, and Dr. Yildiray Yildirim for serving on my defense committee. I am especially grateful to Dr. Yildirim for serving as committee chair on my defense.

I owe many thanks to my parents, Mustafa and Elife Uyar, for their continuous moral support and prayers. I thank my wife and children for their patience during the course of my work toward the Ph.D. degree.

I would like to express my appreciation to many friends. Of these, Ozgur Balsoy deserves special thanks for his passionate assistance in many occasions. I would also like to thank Mehmet Sen, Gurhan Gunduz and Mustafa Varank for their company and support.

Chapter 1

Introduction

Increasing computing power and network bandwidth provide new opportunities for distant communications and collaborations over Internet. In addition to telephony style point-to-point communications, Internet makes it easier to develop multiparty videoconferencing systems for group communications. These systems remove the constraint for physical proximity to have meetings and make it possible for people to meet online from all over the world. They provide real-time audio and video delivery services among meeting participants with additional features such as chat, file sharing, shared display, etc. The videoconferencing systems are used in many different ways from telemedicine to distance education [VC-CB]. However, they have been most popular in education and business sectors. AccessGrid [AG] and VRVS [VRVS] enable researchers around the world to collaborate in projects by saving travel time and expenses. H.323 [H323] based systems make it possible for companies to conduct real-time business meetings with remote employees and clients. In an increasingly global

economy, videoconferencing systems play a vital role to connect the globally distributed workforce of worldwide companies [FRIEDMAN].

The usage of videoconferencing systems are increasing even more with the increase on the number of broadband enabled devices and multimedia capable endpoints. In addition to homes and small offices, even cell phones will have broadband Internet access in the near future with the implementations of 3G [3G] standards. Furthermore, many inexpensive audio and video gadgets are produced for end users. There are many USB headsets with excellent voice quality and many USB webcams with excellent video quality. Many cell phones and PDAs are also equipped with audio and video capabilities. Therefore, the number of end users that are able to attend videoconferencing sessions is increasing rapidly. This requires universally accessible videoconferencing systems that support diverse set of users including cell phones, PDAs, PC based systems, room based systems, etc. Furthermore, increased number of users requires scalable videoconferencing systems that can deliver thousands or tens of thousands of simultaneous audio and video streams. In addition to audio and video delivery, such systems should provide scalable media processing services such as media transcoding, audio mixing, video merging, etc. to customize the media streams according to the needs of lower end clients.

However, developing videoconferencing systems over Internet is a challenging task, since audio and video communications require high bandwidth and low latency. In addition, the processing of audio and video streams is computing intensive. Therefore, it is particularly difficult to develop scalable systems that support high number of users with various capabilities. Current videoconferencing systems can not fully address the

problem of scalability and universal accessibility. IP-Multicast [ALMEROTH] based systems such as AccessGrid requires multicast enabled endpoints. This limits it to high end privileged users. On the other hand, H.323 based systems lack flexible and scalable architecture to support high number of users with diverse capabilities. We believe that with the advancements in computing power and network bandwidth, more flexible and service oriented videoconferencing systems should be developed.

While videoconferencing systems have been developed with an emphasis on high performance and bandwidth savings, other forms of content distribution systems have emerged to provide Internet scale message distribution mechanisms. Particularly publish/subscribe systems [BANAVAR, EUGSTER, BALDONI] have evolved from earlier forms of distributed messaging paradigms to provide loosely coupled messaging middleware to facilitate group communications with richer interactions. They provide an asynchronous and scalable group communication infrastructure that is also suitable in principal for videoconferencing systems. Event producers and consumers are decoupled completely. Event producers publish their messages on topics on the broker network and consumers receive them by subscribing to those topics through the broker network. Event producers need to know neither the identity nor the number of receivers. Similarly, consumers do not need to know anything about the producers. Once a message is published on a topic, the broker network is responsible for delivering it to all subscribers. These message delivery systems can be exploited to deliver audio and video streams in videoconferencing sessions.

Today there are many publish/subscribe systems both from research and industrial community. Some of the well known research projects are NaradaBrokering

[NB1] at Indiana University, SIENA [SIENA] at University of Colorado, and Gryphon at IBM Research [GRYPHON]. Some of the well known industrial solutions are SonicMQ Application Server, IBM WebSphere Application Server, BEA WebLogic Server, Sun Java System Application Server, and JBoss Application Server.

There are many benefits of using publish/subscribe middleware solutions for videoconferencing systems. We can enumerate some of the benefits as follows:

- They can provide a unified message delivery middleware. In addition to real-time audio/video delivery, they can deliver messages for all collaborative applications. This reduces overall system complexity tremendously.
- Their reliable group communication mechanism provides a flexible middleware to implement scalable media processing and meeting management services.
- They can go through firewalls, NATs and proxies.
- Their security infrastructure can be used to implement videoconferencing security services.
- The performance monitoring services can be utilized for audio/video conferencing systems.

1.1 Motivation and Implications

Although it is very difficult to predict the impact of a new technology, we can envision many cases in which a scalable and universally accessible videoconferencing system offers significant benefits. In education, such a system would make it possible

for a university to offer hundreds or thousands of simultaneous online classes with audio and video feeds. Students could join the classes with a variety of devices from all over the world. In addition to formal classes, such a system could offer new opportunities for students to socialize and work in groups. For businesses, such a system could be used in many different ways such as product advertisements, employee trainings, business meetings, etc.

Today there is no videoconferencing system that can provide such services. H.323 based videoconferencing systems that are provided by such companies as Polycom [POLYCOM] and Radvision [RADVISION] can not scale and they are very expensive. On the other hand, IP-Multicast based systems can not be accessed by low end users such as broadband enabled homes and offices and can not go through firewalls. Therefore, a scalable and universally accessible videoconferencing system is necessary to meet the ever increasing needs of online multiparty communications.

1.2 Research Issues

In this thesis, we investigate the question of how to develop scalable and universally accessible videoconferencing systems over Internet. We propose a novel architecture for videoconferencing systems by utilizing the researches in the areas of publish/subscribe systems [BANAVAR] and service oriented computing [SOA1, SOA2]. This architecture is based on the clear separation of major tasks in videoconferencing sessions. It identifies the common tasks performed in videoconferencing sessions and provides independently scalable components for each task. We identified that there are three main tasks performed in videoconferencing sessions: audio/video distribution, media processing and meeting management. We

propose using publish/subscribe event broker systems for the distribution of real-time audio and video streams. We also propose a service oriented architecture to provide media processing and meeting management services that are scalable, flexible and universally accessible.

There are two major research issues that we address in this thesis. The first one is the investigation of using publish/subscribe systems for the delivery of real-time audio/video streams in videoconferencing sessions. We use NaradaBrokering [NB1, NB3] publish/subscribe event broker network to implement and test the ideas presented in this thesis. It is an open source project and provides scalable architecture and many additional features such as reliable message delivery, comprehensive security framework, performance monitoring infrastructure, flexible transport architecture. The second one is the investigation of using service oriented architecture for videoconferencing systems. Here we identify the research issues in more detail under these two main categories.

1.2.1 Using Publish/Subscribe Systems for the Delivery of Real-time Audio/Video Streams in Videoconferencing Sessions

Using publish/subscribe systems for the delivery of real-time audio/video streams is a novel idea and there are a number of issues to be addressed. Although publish/subscribe systems are suitable for implementing videoconferencing sessions in principle, they are not designed to deliver real-time audio and video streams. They lack some of the very important aspects of multimedia communications such as unreliable message delivery and compact data encapsulations. Therefore, some additions and

modifications need to be done to support real-time audio and video delivery in publish/subscribe event broker systems. We identify the research questions as follows:

1. Since real-time audio and video delivery requires low latency, and it can tolerate some package losses, the possibility of using unreliable package delivery mechanisms should be investigated for the best performance.
2. In publish/subscribe systems, messages are exchanged through topics and topics are usually chosen to be strings. However, audio and video streams are composed of many small size packages and the sizes of audio and video packages might increase significantly when string topic names are added. This may result in unacceptable bandwidth increase for audio and video streams. Therefore, the design of a compact topic should be investigated.
3. In publish/subscribe systems, each message tends to have many headers pertaining to content distribution, reliable delivery, priority, ordering, and distribution traces among others. These headers increase the sizes of packages significantly and some of the services provided by these headers are not necessary for the delivery of audio and video streams. Therefore, the design of compact message types for the encapsulation of media packages need to be investigated.
4. Today majority of videoconferencing systems use RTP for the delivery of audio and video streams over Internet. Since the newly designed system will have its own message type for audio and video packages, it will not be compatible with these systems. The ways to interoperate with current videoconferencing systems need to be investigated.

Traditionally real-time audio and video traffic is delivered either using IP-Multicast or hardware based servers, because of their low latency and high bandwidth requirements. However, today ever increasing network bandwidth and computing power make it possible to use software based systems for the delivery of real-time audio and video streams. Therefore, we are proposing to use a software based publish/subscribe messaging system for the delivery of audio and video streams in videoconferencing sessions. Nonetheless, as far as we know, there is no detailed study of the performance and the scalability of software based audio and video delivery systems in the literature. Therefore, it is essential to investigate the performance and the scalability of this software based messaging middleware. First, the capacity of a single broker should be examined in detail, since they are the building blocks of the broker network. Then, the behavior of the broker network should be examined both for multiple smaller size meetings and for single large size meetings. Here we identify the following research questions for the performance analysis:

5. First of all, the factors that affect the scalability and the performance of a broker should be identified and the effect of each factor should be investigated. We identified four factors to investigate: (1) The size of audio/video packages, (2) the frequency of audio/video packages, (3) the number of outgoing streams from a broker, (4) the number of incoming streams to a broker. This will help us understand the performance of a broker in various settings and makes it possible to predict the behavior of the broker network for settings that are not tested.
6. The performance and the scalability of a broker should be tested for real videoconferencing meetings with actual audio and video streams. The amount

of overhead introduced by the broker should be examined to determine the quality of the media delivery. The maximum number of supported users in single and multiple meetings should be determined.

7. In videoconferencing applications, audio communication is the fundamental part of the session. The video communication is complementary and improves the quality of the communication [ISAACS]. However, the video streams tend to be much more bandwidth intensive. Therefore, the effects of video delivery on audio delivery should be investigated. If necessary, the priority should be given to audio stream routing to provide best audio quality.
8. The performance and the scalability of the broker network should be investigated in distributed settings with multiple brokers. Particularly the cost of going through multiple brokers should be examined to provide scalability. The guidelines should be determined to utilize the broker network resources optimally in distributed settings.
9. Lastly, wide area tests need to be performed to investigate the viability of videoconferencing sessions among geographically distributed clients. The overheads of transmission and routing should be examined.

1.2.2 Using Service Oriented Architecture for Videoconferencing Systems

Another difficulty in current videoconferencing systems is the lack of flexible meeting management and service distribution architecture. Internet scale videoconferencing systems require scalable and fault tolerant meeting management

frameworks to handle high number of meetings and participants. In addition, increasingly diverse multimedia endpoints require customized audio and video services. Therefore, scalable media processing services are needed with a flexible architecture to grow dynamically. Using a publish/subscribe system provides new opportunities to develop scalable meeting management and media processing frameworks. We identify the following research questions:

10. First of all, various types of online multiparty meetings need to be examined and the ways to utilize topics need to be determined. Is it best to use a single topic for each meeting, or is it better to use multiple topics for each meeting, one topic for each media stream?
11. What kinds of components should there be in the system to manage the meetings? How will users schedule, discover and join meetings? What kinds of services will these components provide?
12. How will various components in the system communicate with each other? How will they utilize the underlying publish/subscribe system and use topics to interact with one another?
13. What kinds of media processing services will be provided and how will they be distributed among multiple available media processing units?

We will try to answer all the questions that we posed in this section along the way in the thesis and we will also summarize the answers at the conclusion chapter.

1.3 Organization of the Thesis

In the next chapter, we evaluate current videoconferencing systems. We first determine the criteria for evaluation. We determined the following criteria: (1)

scalability, (2) security, (3) traversing through firewalls, proxies and NATs, (4) supporting heterogeneous clients, (5) ease of development, maintenance, usage, and (6) supporting data conferencing. We evaluate each videoconferencing system based on these criteria.

In the third chapter, we give an overview of the proposed architecture of GlobalMMCS. We first outline the design principles and then provide the overview of each component in the system. We also evaluate this proposed architecture based on the criteria set out in the second chapter.

In the fourth chapter, we introduce NaradaBrokering event broker network. We first give an overview of this publish/subscribe messaging system and provide the rationale to use it for real-time audio/video delivery. Then, we explain the modifications and additions that we made to this system to support real-time audio/video delivery.

In the fifth chapter, we provide the analysis of the extensive audio/video delivery tests for NaradaBrokering. We first evaluate the performance of a single broker thoroughly. Then, we evaluate the performance of the broker network in distributed setting. We conducted performance tests for both large size meetings and many smaller size meetings. We also performed some wide area tests in which the clients and brokers were distributed in geographically distant locations.

In the sixth chapter, we present the meeting management architecture and services. We first give an overview of various online meeting types. Then, we present service oriented architecture for meeting and media processing management. We give

the details of the current implementation of this framework. We also provide some performance results for audio and video processing components.

We provide the concluding remarks and the future directions in the seventh chapter.

Chapter 2

Related Work

With the advent of Internet, many researchers and companies realized the opportunities Internet has provided for videoconferencing. A lot of researchers worked on developing a set of protocols for IP-multicast and its deployment. Some telecommunication and internet organizations developed standards to provide a videoconferencing and IP-telephony service to business community and home users. Today there are two well established and one emerging standards and many systems built around them to provide online audio and video conferencing. First we determine the criteria for a videoconferencing system. Then we evaluate these standards and systems based on these criteria.

2.1 Criteria for Videoconferencing Systems

We determined that a conferencing system should have the following characteristics:

1. **Scalability**: The architecture of a videoconferencing system should be flexible to support both small and large number of users at a time. On one hand, a small organization should be able to configure such a system to use for small scale meetings. On the other hand, larger organizations should be able to deploy a distributed version of this system to provide services to higher number of users. This system should allow both very large scale meetings with thousands of users at a time, and hundreds of small scale meetings with thousands of concurrent users. We can envision a virtual university in which hundreds of classes going on simultaneously with thousands of students around the world or we can imagine a virtual graduation ceremony with thousands of students. Such a system should be able to serve this virtual university.

2. **Security**: Security is a very important part of any collaboration system. A videoconferencing system should provide all standard security requirements.

a. **Authentication**: Users should be authenticated to prove their identity.

b. **Authorization**: They should be able to access only those sessions which they are supposed to.

c. **Integrity**: The media streams should not be modified during the transmission by unwanted parties.

d. **Privacy**: The sessions must not be accessed by unwanted users.

e. **Non-repudiation**: The sender of a communication should not be able to deny that they were the sender.

Without providing a secure environment, many organizations will be unwilling to use a videoconferencing system.

3. **Traversing through firewalls, proxies and NAT**: Today many organizations use firewalls, proxies and NATs. Any videoconferencing solution should pay a very close attention to go through these obstacles. Although it would be hard to go though all the firewalls and proxies in the world, at least the system should be designed to go through most of the firewalls and proxies.

4. **Supporting heterogeneous clients**: There are many kinds of multimedia endpoints with different characteristics, such as PC clients, cell phones, PDAs, IP phones, room based high end systems, etc. Their network connection speeds and endpoint capabilities differ greatly. While some of them have low bandwidth connections, others have very high speed Internet access. Some clients might have limited capacity to process the media streams such as IP phones. They might be able to receive only one audio stream and no video stream. On the other hand, some other endpoints can process many audio and video streams simultaneously. For example, an AccessGrid node can receive and process tens of audio and video streams at a time. Therefore, a videoconferencing system should be able to provide services to a diverse set of endpoints. While feeding one audio stream to an IP phone, it should be able to feed multiple audio and video streams to a high end participant.

5. **Easy to develop, maintain and use**: Such a system should have a clear architecture with a clear separation of functionalities. It should be easy to add new services and components. It should also be easy to understand and develop code for that.

6. **Support for data conferencing**: Many audio/video conferencing systems are used along with some data conferencing applications such as whiteboard, chat,

application sharing, etc. These applications make the experience of remote collaboration much richer. Therefore a videoconferencing system should provide or be flexible to get integrated with a data conferencing system.

2.2 Videoconferencing Standards and Systems

Currently there are videoconferencing systems based on two major standards; IP-Multicast [ALMEROTH] and H.323 [H323]. SIP [SIP] is also another emerging standard. Here we give a brief overview of these standards and systems built around them.

2.2.1 Multicast

Multicast [ALMEROTH] is a set of transport level protocols which provides group communications over IP networks. Some of these protocols are PIM-SM, PIM-DM, IGMP, MSDP, BGMP and MADCAP. These protocols are implemented in routers and specify the group formations and management, package delivery mechanisms, inter-domain interactions, etc. Similar to UDP, Multicast provides a best effort package delivery service, but contrary to UDP it delivers packages to a group of destinations, instead of one destination. The groups are formed dynamically, transparent to the user. Each group is identified by a virtual IP address and a port number. When a user sends a package to the group IP address and port number, all participants receive it. The sender does not need to know anything about the receivers; routers are responsible for delivering the data to registered destinations. Therefore, multicast is very well suited for videoconferencing [HANDLEY]. It is particularly suitable for high bandwidth users since they are able to receive many media streams

concurrently. In such a meeting, there is no need for intermediary servers. All participants send their audio and video streams to the group address and routers deliver them to everybody in the meeting. On the other hand, this solution does not work well for low bandwidth or low end clients, since they can not receive or process many concurrent media streams. For these clients, a server side component should process the media and send a customized version.

Multicast is a relatively mature technology. It was first used in 1992 [CASNER] and since then it came a long way. There is a lot of research to improve the Multicast standards and expand the deployment of multicast supporting routers and intranets. Two major problems for multicast are scalability and manageability [ALMEROTH]. But these problems are not usually visible to end users. Currently it works well unless there is a heavy traffic. For end users, firewall traversal and security problems are more serious. Almost all firewalls block the multicast traffic. Therefore special arrangements must be made to let the multicast traffic go through. In addition, current multicast is vulnerable to denial-of-service attacks. Since anybody can join any meeting at any time, malicious users can send some junk streams to disturb a meeting. Although, there are some suggested solutions [MSEC1, MSEC2] for multicast security problems, they are not implemented. It would take years to standardize these solutions and put them on routers, since they are spread all over the world. Moreover, multicast is not widely supported. Even some of the universities in US do not support it. Private corporations usually chose not to support it. Cable modem companies which provide broadband connections to homes and small offices don’t support it either. For dial-up users there is

no hope of getting multicast service. Therefore, it is very hard to use multicast for applications which are expected to serve all Internet users.

Today the most successful videoconferencing system built on Multicast is AccessGrid [AG]. It is a room based videoconferencing system which provides group-to-group communications for institutions with high end network connections. Each site has a room with multiple projectors that project many video streams received from remote sites in large screens. This room has multiple video cameras placed in various points to send multiple video streams from unique perspectives. By providing many video streams from one site and by projecting many video streams in large screens, AccessGrid imitates the real world face-to-face meetings. It tries to provide as much information as possible to create a sense of proximity and closeness. Today there are more than a hundred AccessGrid sites in US and around the world. Most of them are educational and research institutions.

AccessGrid uses one multicast IP address and one port pair for video streams and another multicast address and a port pair for audio streams. Everybody in a meeting sends their audio streams to the group audio address and video streams to the group video address. Then everybody receives all audio and video streams in a meeting through these group addresses. Since AccessGrid uses multicast, it can scale well.

The Access Grid 2.0 utilizes the Globus Toolkit mechanisms [GLOBUS-SEC] for authentication and user identification [AG-SEC]. They encrypt RTP streams with a common key. If someone does not have this key, that person can still get the data but can not decrypt it. Although this solution prevents unwanted users playing the media

streams, it is vulnerable to denial of service attacks. One can easily send malicious traffic to a meeting address and overwhelm the bandwidth of participants.

AccessGrid does not provide any mechanism for going through firewalls and proxies. Each site should negotiate with its network administrators to let the multicast traffic to go through.

Since there is no media processing units in AccessGrid system, all participants receive all audio and video sent by all meeting members. This puts a lot of demand on user network connection. Low bandwidth users can not attend the meetings. In addition, multicast support requirement limits the number of eligible users considerably.

AccessGrid is fairly easy to use and understand, since there is not much server side components.

AccessGrid provides a chat application for node operators to communicate in case there is a problem on voice communications. It also provides a distributed power point application which allows showing power point presentations to remote sites. A whiteboard application is also provided to make remote drawings. In addition, it is also possible to use a commercial web conferencing system in AccessGrid meetings such as Webex.com and Placeware.com. However, these applications do not use Multicast to deliver the messages among users. They require reliable package delivery, but Multicast does not provide reliable transport services. Therefore, they use some other means to implement these services such as dedicated servers for these applications: chat server, power point server, etc.

2.2.2 H.323 overview

H.323 [H323] is a videoconferencing recommendation from International Telecommunications Union (ITU) for package based multimedia communications systems. It defines a complete videoconferencing system including audio and video transmission, data collaboration and session management. Since H.323 is developed by telecommunications industry, it also provides extensive support for traditional telephony and ISDN users to participate in real time meetings through gateways. [TOGA1, TOGA2] gives a good overview of full H.323 system.

H.323 supports two types of multiparty conferencing, decentralized and centralized [H243]. In decentralized model (Figure 2-1), there is no server in the middle. Instead every user sends/receives audio and video directly to/from all other participants in a session. This model can be implemented on top of unicast UDP or multicast. When implemented using UDP, this solution does not scale well. Every user should be connected to every other user in a meeting. It requires a full mesh connection structure. This means that one user sends (n-1) instances of the same audio and video streams when there are n participants in a meeting, resulting in significant bandwidth increase. Therefore this solution can only be used for very small meetings. When it is used with multicast, it can scale well, but Multicast has its own disadvantages as it has been discussed in the previous section.

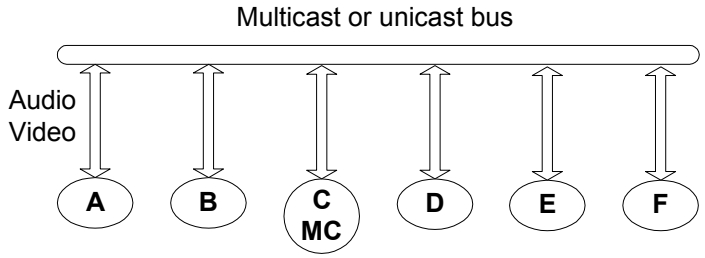

Figure 2-1 Decentralized Multipoint Conferencing.

In centralized model (Figure 2-2) Multipoint Control Unit (MCU) acts as the central server. Its main functions are to negotiate the link between the client and itself, receive and delivery audio and video data, and provide media processing services such as audio mixing, video mixing, and audio and video transcoding. All participants connect to the MCU and negotiate the type of media they will exchange, and then they send and receive audio, video and data through it. Today this form of conferencing is much more common than decentralized model. Some of the advantages of centralized conferencing are not requiring multicast support, providing better control over the central component, easy to add new services, convenient to provide common functionalities such as audio and video mixing [VC-CB].

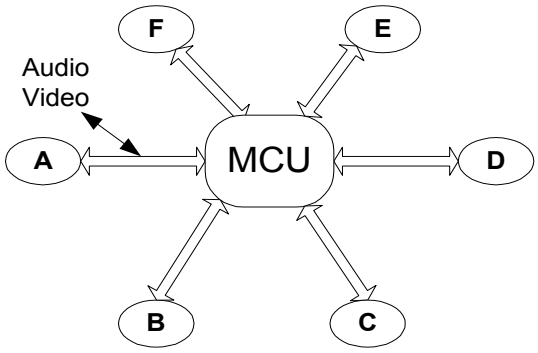

Figure 2-2 Centralized Multipoint Conferencing.

H.323 also suggests a hybrid model (Figure 2-3) in which some participants connect to the central MCU using multicast. This solution scales better than the centralized conferencing approach and more flexible. But in the absence of multicast support, it is identical to the centralized model.

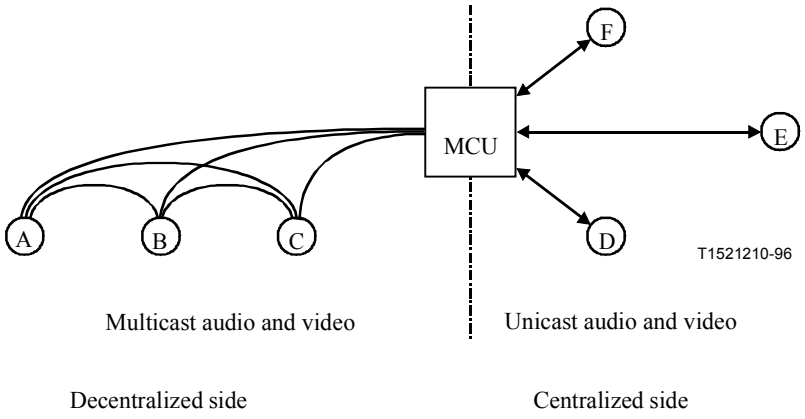

Figure 2-3 Mixed multipoint conference.

Although H.323 does not provide any mechanism to develop a distributed scalable MCU, it realizes the fact that for some large scale meetings a central MCU or the other suggested solutions might not be sufficient. Therefore it defines a mechanism to connect multiple MCUs together to support more participants. This is called MCU cascading. An MCU is connected to another MCU as if it is a client. One of the MCUs is dedicated as the master and the other as the slave. Therefore, some participants connect to the slave MCU and some others to the master. This helps providing services to more participants. But the biggest disadvantage of this solution is that since one MCU is connected to another as a client, only one meeting can be held using multiple MCUs. Therefore it limits the flexibility of this architecture considerably. A scalable architecture must support many simultaneous conferences. Furthermore, this solution is

not very easy to configure. Today's video conferencing systems don't handle MCU cascading transparently. It requires administrator intervention to setup cascading. One meeting should be created on the slave MCU and another meeting should be created on master MCU. Then the meeting at the slave MCU must join the meeting at the master MCU as a client.

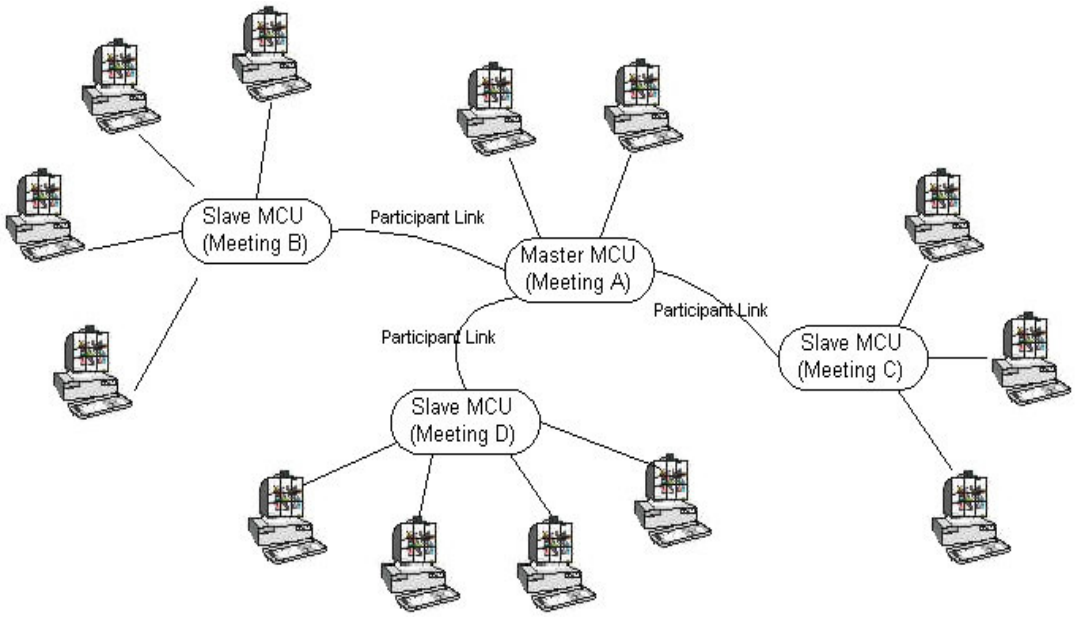

Figure 2-4 H.323 MCU cascading architecture.

It is also possible to connect more MCUs as slave MCUs to one master MCU (Figure 2-4). In this case although we get more scalability, we get an increasingly complex system. Furthermore, it is even possible to setup a multilayer master slave relationship in which one MCU can be a master to its slave MCU while being itself a slave MCU to its master MCU.

As it can be seen, MCU cascading is not really designed to provide a scalable solution, but rather an add-on to already designed system. It can be seen as a quick fix for an occasional need of an organization.

We should note the fact that H.323 defines two logical components for an MCU; Multipoint Controller (MC) and Multipoint Processor (MP). MC handles the media negotiation and session management part and MP handles the audio, video and data communications and processing part. Although, this is a very useful distinction, we believe that MP should also be divided into two logical components to provide a scalable solution. These are media delivery component and media processing component. Media delivery component should handle receiving media from clients, and routing this media to appropriate destinations. On the other hand, media processing component should handle audio mixing, video mixing, media transcoding etc. These two components should be implemented as separate entities and they should be scalable independent of one another. Additional media processing or media delivery components could be easily added to this architecture without affecting the other part.

2.2.2.1 H.323 Based Systems

There are many H.323 based video conferencing systems on the market today. Many of them are hardware based MCUs and support only small scale meetings for 10-20 participants at a time. For example, Polycom recommends MCU cascading to support large scale meetings. CUseeMe provides an interesting architecture. It is software based distributed architecture. We give a brief overview of this architecture.

2.2.2.2 CUseeMe and First Virtual Communications (FVC)

CUseeMe [CUSEEME1, CUSEEME2] or FVC conference server [FVC1, FVC2] provides two types of distributed architecture [FVC3]. One is H.323 cascading which is explained above. The other (Figure 2-5) is connecting many servers using Multicast. Each H.323 endpoint is connected to a conference server and sends/receives media streams through it. This conference server uses multicast to exchange the media streams with other conference servers in the same meeting. In addition, these servers handle audio and video processing functions such as audio mixing, video mixing and transcoding.

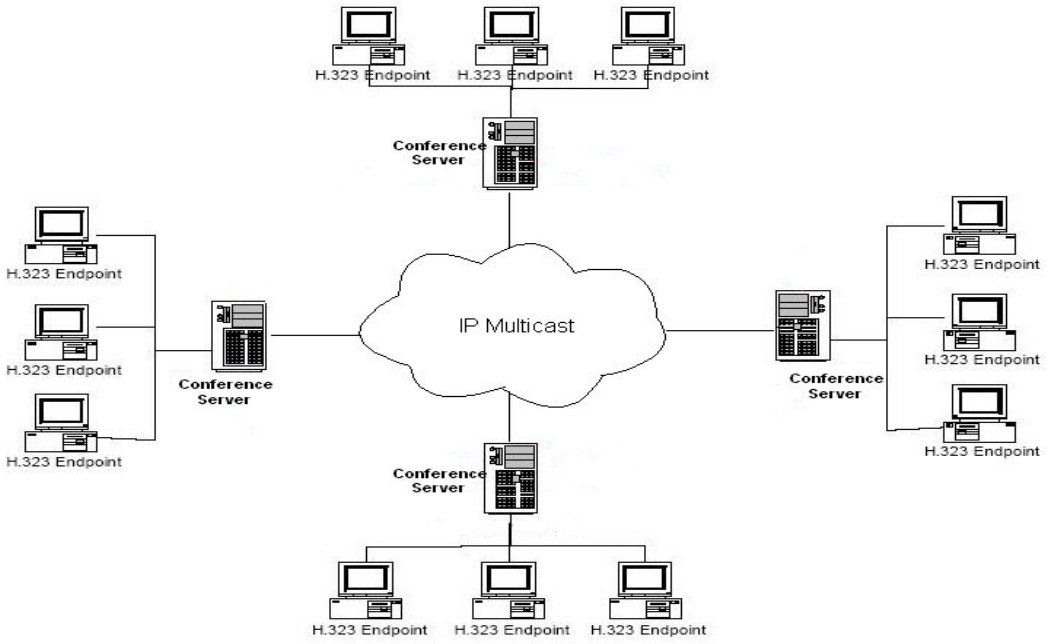

Figure 2-5 CUseeMe Conference Server Architecture.

Although this is much better than cascading, they don't have a clear separation of media processing from media delivery. In addition, since they use multicast to

deliver audio and video streams among servers, it requires multicast support. This limits the deployment of this system to only multicast enabled environments.

2.2.2.3 Other aspects of H.323

H.235 [H235] defines the security mechanisms for H.323 conferencing. It provides authentication, privacy and integrity services. It is a fairly mature and extensive architecture. But most of the current H.323 based systems don't implement this security recommendation.

H.323 based systems are not firewall friendly. Since H.323 uses dynamic ports for media exchange, it requires almost all UDP ports to be open. This defies the purpose of a firewall. However, there are some H.323 friendly firewalls. They examine H.323 messages and open the ports dynamically. These firewalls tend to be more expensive. It is also not very easy for an organization to move from one firewall to another to support videoconferencing.

H.323 based systems provide limited support for heterogeneous clients. They can provide media processing services on the server to accommodate an individual user according to its special configurations. For example, while a low end client can be fed with a G.723 audio stream and no video, a high end client can be provided with many audio and video streams in a session. However, the number of services they can provide and the number of users they can support is very limited, since they do not provide scalable media processing services.

H.323 video conferencing is a complex system. It has many details and covers a wide range. It is not very easy to understand and develop services.

H.323 specifies a complete data conferencing protocol, T.120 [T120]. It defines whiteboard sharing, file transfer and an application sharing mechanism. With application sharing, any application running on a user's desktop can be shown to all other participants in a meeting. Even the control of the application can be passed to a remote user. Although this is very useful, it requires a lot of computing power and bandwidth on the client machine. In addition, it puts extra load on the MCU, since MCU handles the transfer of still images of shared application to other users. Therefore it limits the scalability of MCU considerably.

2.2.3 Session Initiation Protocol (SIP)

SIP [SIP] is a session management protocol from Internet Engineering Task Force (IETF). It can be used to implement videoconferencing systems. It is designed to discover other endpoints and to negotiate the characteristics of a real time session. It is a relatively new protocol and many existing H.323 compatible video conferencing systems support SIP. SIP is more flexible than H.323 and one can easily implement a centralized or decentralized [ROSENBARG] video conferencing solution based on it. SIP does not propose any architecture to implement a video conferencing system. It leaves to the developer to come up with their own architecture.

2.2.4 Virtual Room Videoconferencing System (VRVS)

Virtual Room Videoconferencing System [VRVS, VRVS2] is a mature videoconferencing system from Caltech CMS group for High Energy and Nuclear Physics (HENP) communities. The project started in 1995 and the first production version was released in 1997. Since then, it has been used by thousands of researchers

around the world. It provides audio and video conferencing capabilities for PC based systems. It also supports H.323 compatible end points. In addition, it lets PC clients join AccessGrid sessions.

Since it is not an open project we don't know the full details of their architecture. But they have more than 70 reflectors around the world (Figure 2-6) to distribute media streams. These reflectors receive media streams from users and route them to other interested parties using unicast tunnels and multicast. This reflector system is more like a media distribution network. As far as we understand, their architecture does not provide strong support for media processing. They don't provide any video processing mechanism. They only provide audio mixing for H.323 based clients.

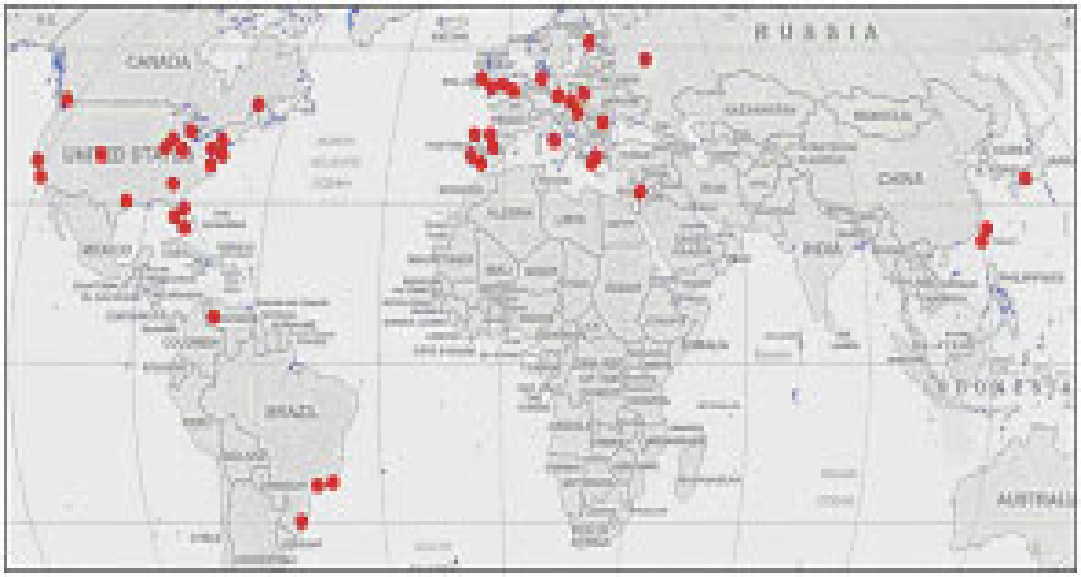

Figure 2-6 VRVS Reflectors around the world.

VRVS requires a reflector to be run behind a firewall, NAT or proxy server to go through these systems [VRVS-SEC]. A user first connects to this reflector inside an

organization and this reflector establishes the connections with other participants in other parts of the network. VRVS reflectors use only one port to simplify going through firewalls. In addition, it can also use a TCP connection instead of a UDP connection to transfer the media. Moreover, in each VRVS client a light VRVS proxy is installed to go through firewalls, NAT, proxy severs and provide security services in the absence of a reflector in an organization. Therefore, VRVS provides a good mechanism to traverse through firewalls, NAT and proxy servers.

VRVS assumes that the connection between the user and the first reflector is secure. It only encrypts the data exchanged among brokers. It uses Data Encryption Standard (DES) to encrypt the media. A VPN connection can also be used between brokers to secure the connection. In addition, VRVS uses a login name/password mechanism to authenticate users. It also sets a password for each meeting.

VRVS provides a chat application, a web browser sharing mechanism and a VNC desktop sharing service. A VNC server runs in the machine of the desktop sharing participant. It sends the images of its desktop to a VRVS sharing gateway which delivers to all other participants in a meeting. Since we don't have the full details of this process, it is hard for us to judge the scalability of their sharing mechanism.

2.2.5 Conclusion

In this chapter, we have developed the criteria for a videoconferencing system and evaluated the current videoconferencing systems based on these criteria. We particularly focused on the scalability of these architectures, since many systems lack a scalable architecture. We showed that almost all systems are good in some aspects but

lacking in other areas. In the next chapter, we present our own solution and evaluate it with these criteria.

Chapter 3

GlobalMMCS Overview

In the previous chapter, we have evaluated the current videoconferencing standards and systems based on the criteria we have set out. In this chapter, after laying out the design principals, we would like to give an overview of our videoconferencing system architecture. We will conclude this chapter by evaluating our approach based on the criteria set out in the previous chapter.

3.1 Design Principles

We consider the scalability as the most important aspect of videoconferencing system architecture. As we have seen in the previous chapter, there are two common ways of providing scalability. One approach is to utilize IP-Multicast infrastructure. Many videoconferencing systems such as AccessGrid, CUseeMe and VRVS use it. Although IP-Multicast is good on scalability, it has other disadvantages such as the lack of widespread support, security and firewall traversal. Therefore we would like to avoid

requiring IP-Multicast in our videoconferencing system design. Some other systems used MCU cascading to provide scalability. However, it is a very limited approach.

We believe that the first step towards building a scalable videoconferencing system is to define and separate the tasks in videoconferencing sessions. Then, independently scalable components can be developed for each task.

We can classify the tasks performed at server side in videoconferencing systems into three major categories:

1. Audio/Video Distribution: This includes delivering audio and video streams from source clients to destinations in real-time. This is a challenging task, since those streams require high bandwidth and low latency. ITU recommends [G114] that the mouth-to-ear delay of audio should be less than 300ms for good quality communication. Therefore, it is essential to provide an efficient media distribution mechanism that will route media streams through best possible routes from sources to destinations. Otherwise, unnecessary network traffic might be generated and additional transit delays might be added. In addition, media streams must be duplicated only when it is needed and never more than one copy of the same stream must be sent on the same link among servers or users. This saves significant bandwidth and provides scalability. The sender publishes one copy of a stream and the distribution network delivers it to all participants by replicating it whenever necessary. Thirdly, since audio and video streams are composed of many small sized packages, minimum headers should be added to all packages. Otherwise, there can be substantial increase in the amount of data transferred.

2. Media processing: There might be many different kinds of media processing performed in videoconferencing systems. The most common ones are audio mixing, video mixing, and media transcoding. Although in a homogenous videoconferencing setting where all users have high network bandwidth and computing power, media processing might not be necessary at server side, it is crucial in videoconferencing sessions where users have various network and device capacities. For example, AccessGrid [AG] does not provide any media processing services since each AG node can receive/send/display tens of audio/video streams concurrently. However, videoconferencing systems that aim to support diverse set of users with various network bandwidths and endpoint capabilities must provide media processing services to customize the streams according to the requirements of users. Some users might have very limited network bandwidth. For those users, multiple audio and video streams should be mixed to save bandwidth, or some high bandwidth streams should be transcoded to produce low bandwidth streams. Some other users might have limited display or processing capacity. For those users, multiple video streams can be merged or larger size video streams can be downsized.

Media processing usually requires high computing resources and real-time output. Therefore, they can limit the scalability of a videoconferencing system severely when implemented poorly. More importantly, they can affect the quality of audio and video distribution if they share the same computing resources with media distribution units. Therefore, media processing units should be separated completely from media distribution units to provide scalability. In addition, it should be possible to add new computing resources dynamically to support high number of sessions with more users.

Moreover, media processing framework should be flexible to allow the implementation of new media processing services.

3. Meeting management: Meeting management includes starting/stopping/modifying videoconferencing sessions. It also includes determining and assigning system resources for these sessions. For example, it includes finding out the right audio mixing unit to be used by a meeting. In addition, it includes the mechanisms for participants to discover/join/leave sessions. Contrary to the media distribution and media processing tasks, session management requires little bandwidth and computing resources. However, it is very important to coordinate and distribute the tasks in such sessions. Therefore, it is very important to design a flexible and scalable session management mechanism.

None of the systems have we examined so far has a very clear separation of functionalities and components. Although H.323 separates session management from media communications, it fails to separate media distribution from media processing. On the other hand, Multicast provides a very good media distribution mechanism but fails to provide a convenient meeting management and media processing mechanism.

3.2 GlobalMMCS Architecture

Global Multimedia Collaboration System (GlobalMMCS) is designed to provide scalable videoconferencing services to a diverse set of users. For each of the tasks outlined above, there is a component in this architecture (Figure 3-1): media and content distribution network, media processing unit and meeting management unit. Meeting management unit creates/deletes and maintains meetings. It provides interfaces

to both users and administrators. Administrators create and delete meetings by accessing the private interfaces of Meeting Manager. Users join and leave meetings or access other media services by using the public interfaces of the Meeting Manager.

We use a unified messaging environment called NaradaBrokering [NB1, NB2, NB3] as the media distribution medium. It is an event brokering system designed to run on a large network of cooperating broker nodes. It is based on the distributed publish/subscribe paradigm and JMS [JMS] compliant.

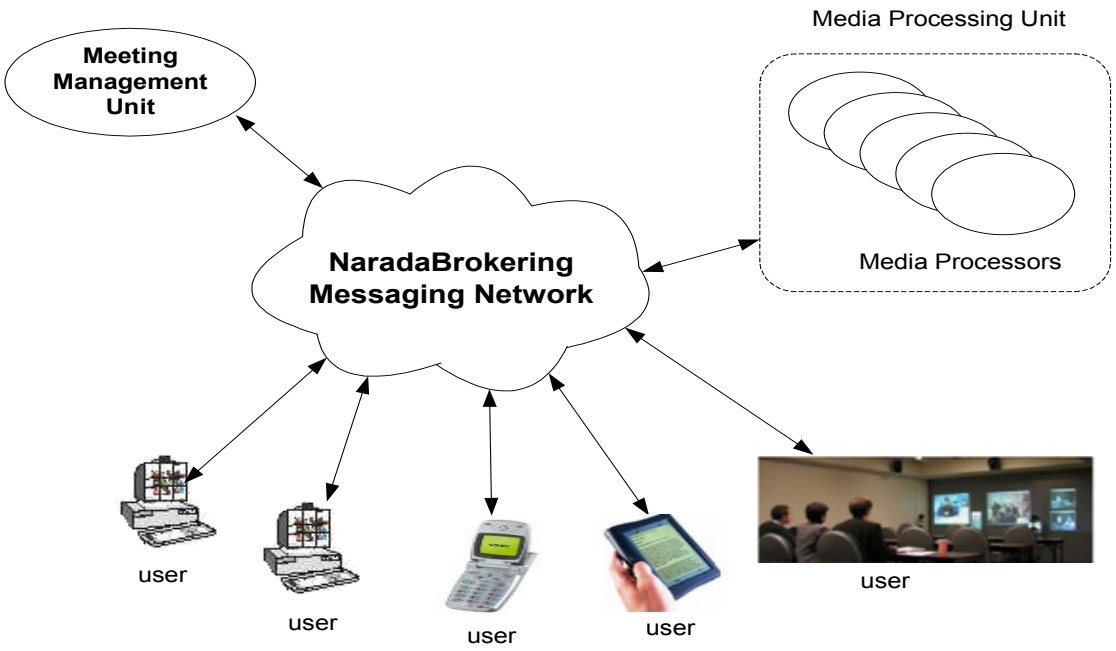

Figure 3-1 Main components of GlobalMMCS architecture

Publish/subscribe messaging systems provide an asynchronous group communications medium. Topics serve as the messaging channels among participants in a session to exchange data. Although, traditional publish/subscribe messaging systems have been used by applications which require reliable message delivery and

rich message feature set, we extend this model to be used by videoconferencing systems.

NaradaBrokering system is very efficient to deliver audio and video streams to a group of destinations in a meeting. The source user publishes one copy of a stream on a topic, and the broker network delivers it to all subscribers by duplicating whenever necessary. It avoids sending multiple copies of the same stream on the same link to save network bandwidth. In addition, by organizing the brokers in hierarchical cluster architecture, it calculates the near optimal routes from sources to destinations.

In addition to media distribution, broker network also handles other message delivery functions. Since it is a JMS compliant system and supports reliable transport protocols such as TCP, a chat application or a distributed power point sharing application can be easily implemented by using the broker network as the group message delivery medium. [WANG] is an effort to implement a distributed shared power point and a shared web browser applications using NaradaBrokering as the messaging middleware. In addition, in our system, all the control messages are delivered through broker network using the reliable transport protocols.

This architecture provides a scalable and flexible framework to distribute media processing units. New computing resources can be added dynamically to increase the capacity of the system and new processing services can be added easily to support the ever changing needs of end users. Moreover, this framework supports multiple instances of media processing units to provide fault tolerance. Currently, we implemented three types of media processing functionalities: audio mixing, video mixing and image grabbing.

NaradaBrokering publish/subscribe messaging middleware provides a very convenient medium to implement media processing units. Each media processing unit subscribes to the proper topic to get the media streams to process, and then publishes back the processed stream on the broker network on another topic. Users access the processed media streams through broker network by subscribing to the topics of processed media streams. This mechanism provides location independence to media processing units by separating them from transmitters and receivers completely. Media processing units do not interact directly with transmitters and receivers. Instead, they only talk to messaging network to receive the streams and to publish the output. Therefore, they can run anywhere as long as they are connected to a broker.

Meeting Manager controls the media processing units. For example, when a new audio meeting is created, meeting manager instructs the Media Processing Unit to start a new audio mixer for that meeting. It also adds users to the mixer as they join the meeting.

3.3 Evaluation of GlobalMMCS

In this section, we would like to evaluate GlobalMMCS videoconferencing system based on the criteria which we set out in the previous chapter.

3.3.1 Scalability

As we pointed out, GlobalMMCS is designed to be scalable from the start. Its separation of functionalities is the key to achieve the scalability. By separating the functionalities and developing independent components, it enables the independent

growth of each component without affecting the others in the system. We provide the performance test results in Chapter 5.

3.3.2 Security

NaradaBrokering has an extensive security infrastructure [NBSEC]. It provides all the standard security requirements. In addition, it provides a key management infrastructure. Each user can be authenticated before publishing or subscribing to any topic. A number of algorithms supported to encrypt messages. One can chose according to its requirements. Each topic has an access control list which allows only the authorized people to access and publish the information. Moreover, it also takes precautions against several security attacks such as denial of service and replay attacks.

3.3.3 Traversing through firewalls, proxies and NAT

One of the advantages of using NaradaBrokering messaging middleware as the message delivery medium is its support for going through firewalls, proxies and NAT. In addition, it supports multiple transport protocols [NBTRANSPORT] such as UDP, TCP, HTTP, SSL, which is essential when going through these obstacles.

We propose two types of firewall traversal mechanisms. In the first approach (Figure 3-2), an NB broker is placed inside the firewall, NAT or proxy, and all clients behind the firewall connect through this broker. This broker then uses either a hole in the firewall or uses a reliable transport protocol such as TCP, HTTP or SSL to connect to the broker network outside. Going through one broker makes it easier for an organization to deal with the issue. It can also provide a mechanism to monitor the media traffic. In addition, a legacy client inside an organization can be supported with

this architecture. That legacy client can use RTP over UDP to connect to the inside broker, and then that broker uses another protocol to traverse the firewall.

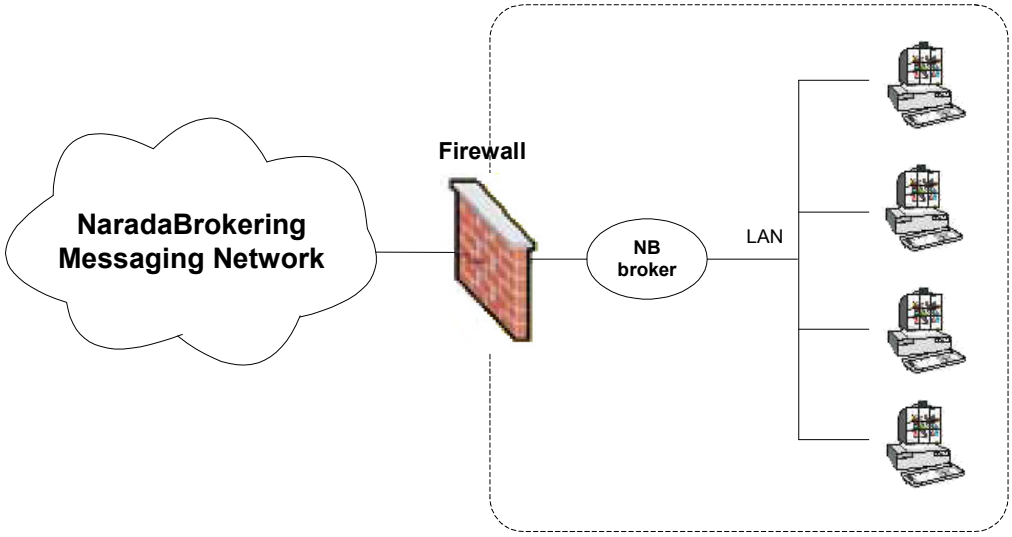

Figure 3-2 Firewall traversal with an NB broker inside an organization

Another way of going through a firewall, NAT, or a proxy server, is to use an NB compliant client which can directly talk to a broker outside the firewall (Figure 3-3). This NB compliant client should use a different transport protocol such as TCP, HTTP or SSL, to be able to go through the firewall. In this second approach, only an NB compliant client can be supported. A legacy client can not join the meetings. This second approach might be preferable for small organizations or occasional use. On the other hand, it would be better to install a broker for large organizations with many users accessing the videoconferencing sessions frequently. Such an organization might also have many internal sessions without any outside participation.

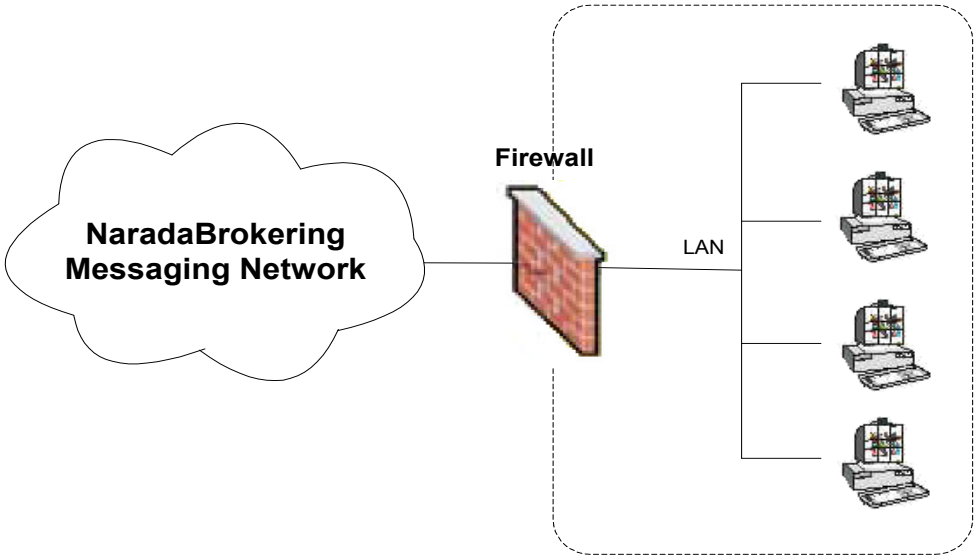

Figure 3-3 Firewall traversal with NB compliant clients inside an organization

Although using a reliable transport protocol is more costly than using an unreliable protocol such as UDP, it is the most convenient way of traversing firewalls. Some companies such as Polycom and FVC have H.323 friendly firewall solutions, but this approach requires an organization to change its whole firewall system with a new one. This is unacceptable for majority of cases. In addition, the cost of using a reliable transport protocol can be minimized if the broker inside an organization and the other outside broker which that broker connected to is placed as close as possible.

3.3.4 Supporting heterogeneous clients

Since we provide a scalable media processing framework, we can support a diverse set of end points. While an AccessGrid node may get all audio and video streams in a meeting by subscribing to all topics of a meeting, a low end client such as a polycom ViaVideo can receive one mixed audio and one mixed video stream. We

give an option to low end clients to select the video streams they want by providing the list of available video streams in a session. Our media processing units perform the audio and video mixing necessary to support these low end clients. In addition to audio and video mixing, we provide an image grabbing service. It receives all video streams in a meeting and gets the snapshots of these streams regularly. These images are presented to users for them to make intelligent decisions regarding which video stream to receive.

3.3.5 Easy to develop, maintain and use

We use publish-subscribe programming paradigm to implement this videoconferencing system. It is a very well understood asynchronous group communication model. It provides well designed abstractions and components for group interactions. We apply these well established standards to videoconferencing applications. Since videoconferencing applications are also group based systems, it fits well to this model. For example, we publish a video stream to a topic on the broker network, and then all interested users receive this video stream by subscribing to this topic. The sender of the video stream does not need to know anything about the receivers. Broker network handle the subscriptions and also delivery of the video stream to interested parties.

This programming model is similar to multicast but it gives more control to the users and administrators. In the current multicast, one disadvantage is that there is no authority which assigns the multicast addresses and port numbers, instead anybody can use any address at any time. In addition, once a stream is published to an address, anybody can receive that audio or video stream provided that that user knows the

address. On the other hand, in our system, since we have full control over the broker network, we assign each topic to a particular use; therefore no other user can use that topic. Moreover, an access control list can be associated with each topic to limit the authorized users to receive the media on that topic. Therefore, while our system provides the ease of use of multicast, it avoids the limitations posed by it.

Our architecture also provides an easy-to-manage environment, since all applications use a unified messaging middleware. Compared to having two different middleware for data and media delivery, this approach makes it much easier to maintain and manage the system. In addition, its clear separation of components makes it very convenient to debug the system.

3.3.6 Support for data conferencing

One of the most important advantages of our architecture is to provide a unified messaging middleware for both media and data applications. Since our distributed messaging middleware supports both reliable and unreliable data delivery mechanism, a data conferencing module can easily be developed without requiring another middleware. In fact, Garnet [FOX] collaboration environment based on JMS is an example of such a system, which provides application-sharing, whiteboard, and shared-display support.

In addition to these evaluated criteria, our system provides a relatively inexpensive solution since it uses regular computers compared to specialized hardware based systems.

3.4 Comparison of Videoconferencing Systems

In the following table, we provide a summary of the features of GlobalMMCS and the other videoconferencing systems that we evaluated in Chapter 2.

Table 3-1 Comparison of videoconferencing systems

	IP-Multicast based Systems	H.323 based Systems	VRVS	GlobalMMCS
Scalability	Scales well.	Does not scale well.	No information available.	Scales well.
Security	Very limited security. Vulnerable to denial-of-service attacks.	Provides security services but most H.323 based systems do not implement it.	Provides acceptable security services.	Provides extensive security framework. However, Limited implementation.
Firewall, Proxy, and NAT Traversal	Almost all firewalls, NAT and Proxies block multicast traffic.	Not firewall friendly. Requires firewalls to let H.323 traffic in.	Good support for firewall, NAT and proxy traversal.	Good support for firewall, NAT and proxy traversal.

Heterogeneous client support	Only high end users are supported.	Limited support for heterogeneous clients.	Limited support for heterogeneous clients.	Extensive support for heterogeneous clients.
Ease of development, maintenance, and use	Very easy to use, and develop video conferencing applications.	H.323 is a complex standard. Not very easy to understand and develop code.	No information available.	Provides well understood communication model. It is easy to understand and develop code.
Data Conferencing Support	Data conferencing applications are implemented as separate services without using multicast infrastructure.	A complete data conferencing protocol is provided.	Limited information. Difficult to judge their architecture.	Provides reliable group communication infrastructure to develop data conferencing applications.

3.5 Conclusion

In this chapter, we have given our design criteria for our videoconferencing architecture. We have also given an overview of our system architecture. Our evaluation shows that our system is based on sound principles and provides an affordable and manageable architecture. In the upcoming chapters, we will give the details on media and content distribution network, meeting management and media processing units.

Chapter 4

Media Distribution Middleware

In this chapter, we cover the media distribution mechanism in our system. First, we point out the challenges media distribution poses and give the current solutions. Then, we introduce NaradaBrokering event brokering middleware and the enhancements that we made to support real-time media stream delivery.

4.1 Requirements for Media Delivery and Current Solutions

As we have pointed out in the previous chapter, media distribution has three main constraints:

1. High Bandwidth: Media streams are bandwidth intensive. Particularly the video streams require very high bandwidth. Therefore, media packages must be replicated only when it is necessary during the transmission. Namely, the transfer of multiple copies of a stream must be avoided on the same link. In addition, the best routes should be chosen from sources to destinations to avoid the extra load on the

network. Furthermore, since media streams are composed of many small sized packages (a few hundred bytes), distribution system should add minimum length headers to media packages in order to avoid generating unnecessary load on the system.

2. Low latency: Since videoconferencing sessions are used for real time meetings, media streams must be delivered to the destinations in a timely manner. When two remote speakers are in communication, they should not feel the latency introduced by the transfer of the audio and video data. This can be achieved by keeping the transmission latency lower than a few hundred milliseconds. If some packages arrive later than the acceptable buffering time period, these packages are considered as lost packages and they incur gaps in the communications. Therefore, the delivery of audio and video streams should be stable during the real-time sessions.

3. Tolerate Package Loss: Audio and video applications can tolerate some package loss during the transmission as long as they do not affect the quality of the communication seriously. This lets us to use unreliable transport mechanisms to deliver audio and video streams.

A lot of efforts have been put into IP Multicast to solve the media distribution problem over Internet. Although Multicast served the needs of many people and institutions, it lacked widespread use and support as we have pointed out in previous chapters. Particularly, its requirement for transport level support from routers discouraged many internet service providers. This led many researchers to look for application level multicast solutions which can be implemented in higher levels and work over the current Internet infrastructure. Overcast [OVERCAST] is such an effort to develop an overlay network to provide single source multicast for media streaming.

It tries to serve recorded media material to large number of users by replicating them on the disks along the path from sources to destinations. End System Multicast [ESM] is another effort to implement multicast services for small groups by using end systems for routing without any middleware servers. Bayeux [BAYEUX] is another project to provide application level scalable multicast services based on Tapestry [TAPESTRY]. All of these systems are heavily influenced by Multicast and tries to replicate the multicast services on application level. We take a novel approach and utilize distributed publish/subscribe brokering systems to distribute the real time media.

4.2 Overview of Publish-subscribe systems

Publish/Subscribe systems have evolved from the previous forms of messaging paradigms to answer the needs of Internet wide applications. While older distributed systems required tightly coupled applications and synchronous communication mechanisms such as message passing, remote method call (RPC), and shared spaces, Internet wide systems required loosely coupled applications and asynchronous messaging paradigms. Publish/subscribe systems addressed these needs by providing a messaging middleware that decouples producers and consumers on time, space and synchronization [EUGSTER, BALDONI]. The decoupling of producers and consumers on time means that consumers do not have to consume messages as they are produced. The middleware can deliver the recorded messages at a later time. For videoconferencing applications this feature can be exploited when archiving and replaying the archives. The decoupling on space means that producers do not have to know anything about the consumers and consumers do not have to know anything

about the producers. This opens up the door to implement scalable videoconferencing sessions. Producers publish only one copy of a stream and the brokering middleware delivers them to all subscribers. The decoupling on synchronization means that producers and consumers do not have to be blocked, when receiving or sending messages. This lets a client to send and receive many streams at a time.

Publish/subscribe systems can be classified into two major categories: topic-based and content-based. In topic-based systems, users exchange messages on shared channels. When a producer publishes a message on a topic, all subscribers of that topic receive that message. It is a many-to-many communication medium. In content-based systems, subscribers specify the kinds of messages they are interested and broker network checks the messages according to user's specification and deliver the message when they match. Topic-based messaging is better suited for audio and video distribution. Since an audio or video stream is composed of many consecutive packages, it is more efficient to publish them on the same topic. It is also more convenient for receivers to subscribe to a topic to receive all the messages belonging to a media stream. Therefore, from now on, we will only focus on topic-based publish/subscribe systems.

Although publish-subscribe systems provide a convenient and scalable environment to deliver audio and video streams, they are not designed to serve real-time multimedia traffic. They are usually used to deliver guaranteed messages by employing reliable transport protocols. In addition, they do not focus on delivering high bandwidth traffic or reducing the sizes of the messages they transfer. It is more important for them to provide more services than saving bandwidth. Each message

tends to have many headers related to the content description, reliable delivery, priority, ordering, distribution traces, etc. Many of these services are not important for audio and video delivery. Therefore we need to design new message types and transport protocols to support multimedia traffic.

We use a distributed event brokering system called NaradaBrokering based on publish/subscribe paradigm to implement our media distribution middleware. First we give an overview of this system and then introduce the additions we have made to support real-time media delivery.

4.3 NaradaBrokering

NaradaBrokering [NB1, NB2] is a distributed publish/subscribe messaging system which organizes brokers in a hierarchical cluster-based architecture (Figure 4-1). The smallest unit of the messaging infrastructure is the broker (small circles in Figure 4-1). Each broker is responsible for routing messages to their next stops and also handling subscriptions. In this architecture, a broker is part of a base cluster (the rectangles in Figure 4-1 such as a, b, c, d, e, etc.) that is part of a super-cluster (the collection of rectangles in Figure 4-1 such as SC1, SC2, SC3, SC4, etc.), which in turn is part of a super-super-cluster (collection of super-clusters in Figure 4-1 such as SSC-A, SSC-B, SSC-C, etc.) and so on. Clusters comprise strongly connected brokers with multiple links to brokers in other clusters, ensuring alternate communication routes. This organization scheme results in the average communication "pathlengths" between brokers that increase logarithmically with geometric increases in network size, as opposed to exponential increases in uncontrolled settings.

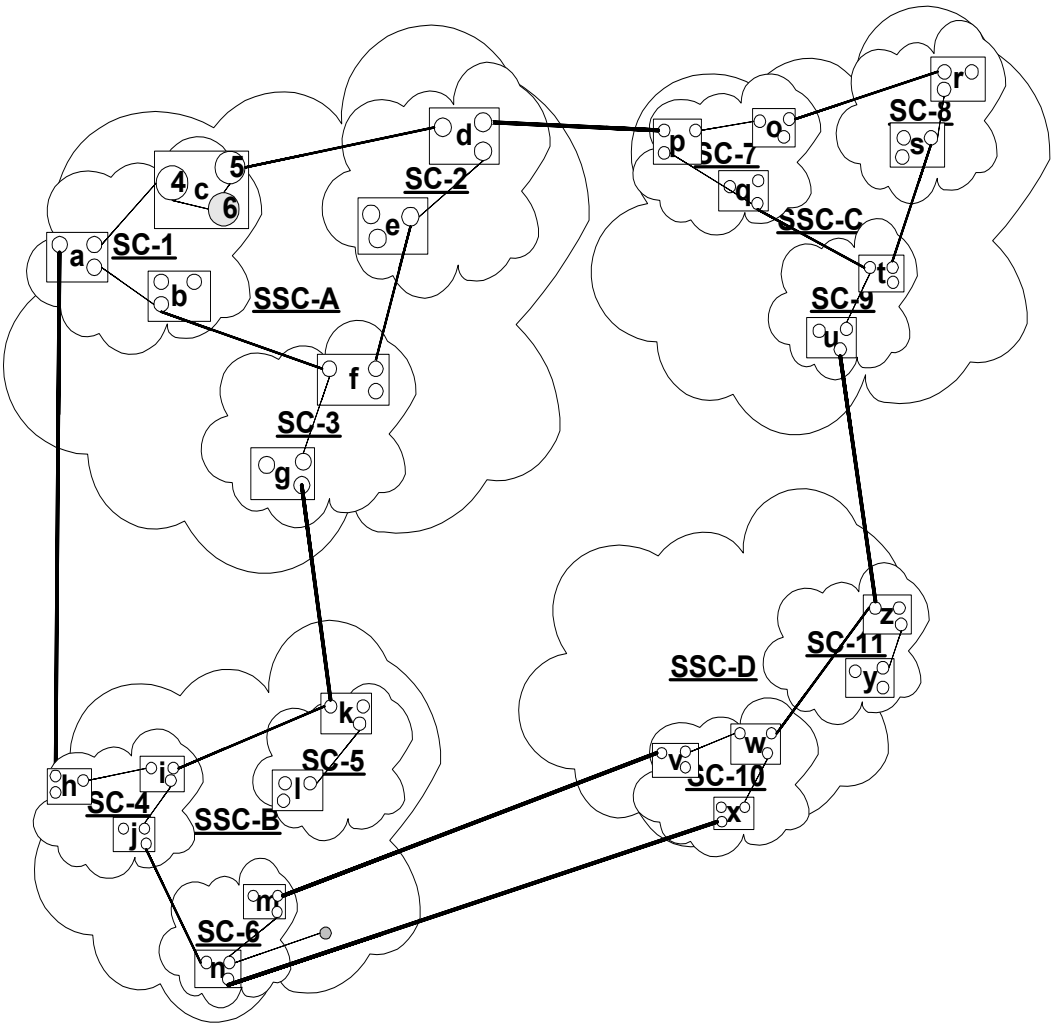

Figure 4-1 NaradaBrokering broker organization

NaradaBrokering supports dynamic broker and link additions and removals. While adding new brokers and links, it implements a broker organization protocol to avoid an unstructured broker network which hampers the development of efficient routing strategies. This protocol makes sure to add new brokers or links into appropriate places in the network to maximize the overall system efficiency.

Each broker keeps a broker network map (BNM) of its own perspective to efficiently route the messages to their destinations with a near optimal algorithm [NB2]. Each broker keeps the state of all brokers and links in the base cluster. Then, they keep the inter-cluster links and relevant brokers for the upper clusters in the network by avoiding the details of those clusters. Although this prevents calculating the best routes for messages to their destinations, it significantly reduces the amount of state information kept in brokers. In addition, the calculation of routes can be made much faster than having all broker state information in all brokers.

Clusters can be constructed either in a traditional sense, groups of broker nodes connected together by high speed links, or geographical proximity, geographically closer brokers form a cluster. In addition, an institution or a company with more than one broker can decide to form a cluster.

Messages are routed only to those routers that have at least one subscription for that topic. This prevents unnecessary message traffic on the system. In addition, near optimal routes are chosen from producers to consumers based on the BNM in each broker. Moreover, messages are routed only to the intended destinations and they are prevented from being routed back to the producers. In the first broker that a message is published, a dissemination trace value is added to the message before forwarding it to the next broker; this value is then updated in each hop to reflect the visited brokers. Each broker checks this value and avoids routing this message back to already visited brokers. This dissemination trace value is removed in the last broker before sending it to the destination clients.

NaradaBrokering has a flexible transport mechanism [NBTRANSPORT]. Its layered architecture makes it easy to add new protocols. In addition, when a message traverses through broker network, it can go through different transport links in different parts of the system. A message can be transported over HTTP while traversing a firewall but later TCP or UDP can be used to deliver it to its final destinations.

NaradaBrokering is JMS compliant and provides support not only for JMS clients, but also for replacing single server JMS systems transparently [NBJMS] with a distributed NaradaBrokering broker network. Therefore, in addition to audio and video delivery, this broker network can also be used for the delivery of reliable messages. Other collaboration applications such as chat, application sharing, etc. can use the same broker network for message delivery.

Another important feature of NaradaBrokering is the performance monitoring infrastructure [GUNDUZ]. The performance of the links among brokers is monitored and problems are reported on real-time. In addition, this information is used to route messages through best paths.

4.4 Incorporating Support for Audio/Video Delivery in NaradaBrokering

Although NaradaBrokering publish/subscribe messaging system provides a scalable distributed architecture for content delivery, it did not provide any support for real-time traffic. Therefore, we needed to add support for that. There are five major additions we have made:

1. Adding support for an unreliable transport protocol

2. Implementing a distributed topic number generation mechanism
3. Designing a Unique ID Generation Mechanism
4. Designing a new compact event
5. Adding support for legacy RTP clients

In addition to these new features, we made some improvements in the routing algorithm of NaradaBrokering system to provide better services and scalability. We will discuss these improvements in the performance test sections in the next chapter.

4.4.1 Adding Support For An Unreliable Transport Protocol

Since audio/video traffic can tolerate some package loss during transmission and it requires timely delivery of data, unreliable transport protocols are used to avoid the extra cost associated with error correction and package retransmission in reliable transport protocols such as TCP. For point-to-point media transfer, UDP is the preferred choice. Therefore, we have added a link implementation based on UDP into the NaradaBrokering architecture. This UDP link enables both client-broker and broker-broker communications.

4.4.2 Implementing A Distributed Mechanism For Topic Number Generation

NaradaBrokering implemented a string based topic mechanism. Although this was very useful for other applications, it was not adequate for media delivery. Since media streams are composed of many small sized packages and they are bandwidth intensive, it is very important to add minimum headers to each message. When strings are used as topic names and a topic name is added to each media package, this may

result in significant increase in the required bandwidth and it adds more load on the brokers and links. When strings are serialized, each character takes at least one byte. Depending on the size of the string topic name, tens of bytes can be added to each message. Since media packages can be as low as 20 bytes, it would not be efficient to add tens of bytes to each message as the topic name.

One way of solving this problem is to impose a limit on the size of the topic string. We can require each topic to have at most 8 characters, but this would limit the number of possible topic names significantly. In addition, the collision of topic names would increase. On the other hand, increasing the maximum size of the topics would result in more bandwidth and load, though it would provide more options. Therefore, we have decided to implement a topic mechanism which can provide more options and take less space.

Although one aspect of the topics is their size, another aspect is the way they are created and their uniqueness is insured in a distributed setting. Here we outline three conditions to meet for a distributed topic management system:

1. A topic generator must be able to create topics without interacting with other topic generators in the system. We avoid centralized solutions for fault tolerance and speedy execution.

2. A topic generator should be able to fail and start over without requiring saving its state to a stable storage. Namely, topic generators must be stateless.

3. Each topic should have a predetermined size and their size should be as small as possible.

The first condition requires the spatial independence of a topic generator. This can be achieved by assigning a unique topic generator id to each topic generator in the system. Then this unique id can be added to every topic constructed by that generator to provide system wide uniqueness. In NaradaBrokering network, each broker is assigned a unique id. We can utilize this mechanism to provide unique ids for each topic generator. We require that a topic generator runs in each broker. Clients ask this topic generator to construct a new topic for them. The broker id in NaradaBrokering system is 16 bytes. This is obviously too long to add to each topic. On the other hand, this broker id is not designed to use the minimum space, it is designed to provide the best performance. Therefore, it does not utilize the 16 bytes range efficiently. Instead, it is possible to generate a 20 bit id from this 16 bytes broker id with a simple conversion. This 20 bit is small enough to add to each topic. This way each topic generator in brokers will be able to generate topics independent of other brokers in the system.

The second condition requires the temporal independence of a topic generator. This can be achieved by adding a timestamp value to each topic. Since NaradaBrokering brokers are synchronized [NBNTP] with high accuracy clocks using Network Time Protocol [NTP], it simplifies the problem considerably. This eliminates the conditions where the clock of a computer can be changed backward. With this synchronization mechanism in use, we can assume that the time always flows forward, though it may stall for short periods of time when synchronizing. Therefore, we can add a timestamp to each topic to provide temporal independence of topics without requiring state savings to file system when restarting a broker.

The third condition requires that topics should have minimum size. We can construct a topic number by combining the topic generator id with a timestamp. Since topic generator id is 20 bits, we need to determine the size of the timestamp. If we set the total topic number size as 32 bits (4 bytes), remaining 12 bits would provide 212=4096 different options for the timestamp. Therefore, it is too small. When we set the total topic number size as 64 bits (8 bytes), 44 bits are left for the timestamp. This would provide 244=17592186044416 distinct timestamp values. If we increment timestamp value by one in every millisecond, this would be enough for 557 years. Therefore 64 bits topic number (Figure 4-2) is a good choice. It is small enough to add to each audio and video package.

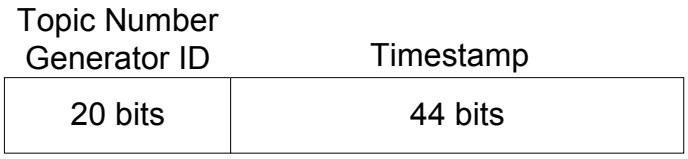

Figure 4-2 Topic number for 8 bytes

In some cases, it is possible that more topic numbers be generated than one millisecond interval. If this is the result of requesting many topic numbers in small amount of time, then the topic number generator in the broker can keep the state of some past period of time and assign the unused topic numbers. But this can not go beyond the last starting time of the broker. Another option for such a client might be to request topic numbers from many brokers instead of one broker.

In addition, for some applications which use a lot of topics, it would be better to get a list of topics with one request instead of sending a request for each topic number.

When allocating a list of topics, topic number generator can not use the numbers from future timestamp values. It can either wait for the time to pass or it can use from the past unused topics.

In conclusion, this mechanism provides a fast and scalable solution to generate unique topic numbers with 8 bytes. Since a broker does not interact with other brokers, or keeps track of the unused topic numbers, it can generate topic numbers very quickly. In addition, this mechanism lets brokers fail and start over without interacting with other brokers or using a stable storage device. Moreover, it guarantees to generate unique topic numbers. UUID [UUID] solves a similar problem with 16 bytes. Nonetheless, the uniqueness of those ids is not guaranteed. There is a small chance of collisions in UUID algorithm.

4.4.3 Unique ID Generation

In distributed systems, components usually need system wide unique ids to distinguish themselves from other components. UUID tries to solve this problem but it can not guarantee the uniqueness of the generated ids. The main reason for that is the difficulty of solving this problem for all applications universally. However, when we limit the distributed application domain to NaradaBrokering brokers as in the case of generating unique topic numbers, it can be possible to generate unique ids efficiently. Therefore, our distributed unique topic generation mechanism can also be used to generate unique ids for distributed components. Either another instance of the topic generator can be run in every broker as an id generator using the same algorithm or the same topic generator can be used to generate unique ids. For those systems that would require heavy uses of topic numbers and ids, it would be better to run the topic

generator and the id generator separately in each broker. Less demanding systems can run only one instance to be used as both topic and id generators.

4.4.4 Designing a New Event

In publish/subscribe messaging systems; messages tend to have many headers, most of them related to the quality of services provided. Since audio and video streams do not require quality of services such as persistence and reliability, many of these headers are unnecessary. For example, a message in JMS API has at least 10 headers. Many of them are redundant in the context of audio and video delivery. These headers take around 200 bytes when they are serialized to transfer over the network. If these 200 bytes are added to each audio and video package, it results in substantial increase in the bandwidth of the audio or video streams. For example, a ULAW audio package for 20 ms has a size of 172 bytes including the RTP header and entails 64 kbps network bandwidth. Padding an extra 200 bytes of header to each audio package results in the bandwidth requirement of up to 148 kbps. Then, there is the cost associated with serializing and de-serializing the multimedia content. Therefore, we need to design a new event type with minimum headers and minimum computational overhead.

We designed a special event, the RTPEvent (Figure 4-3), to encapsulate the media content that comprises of 5 elements. There are two headers identifying the event type. Both headers are 1 byte. Event header identifies the event as RTPEvent among other event types in NaradaBrokering system. Media header identifies the type of the RTPEvent such as audio, video, RTCP, etc. These headers are followed by topic name (8 bytes) encapsulating information about the topic number this event belongs to. To eliminate echo problems arising from the system routing content back to the originator

of the content, information pertaining to the source is also included. This information can be represented as an integer, which amounts to 4 bytes. Finally, there is the media content itself as the payload in the event. Although, in Figure 4-3 an RTP package is seen as the payload, it can be any data type. Therefore, the total length of the headers in an RTPEvent is 14 bytes. 14 bytes is small enough to add to each audio and video package transferred in the system.

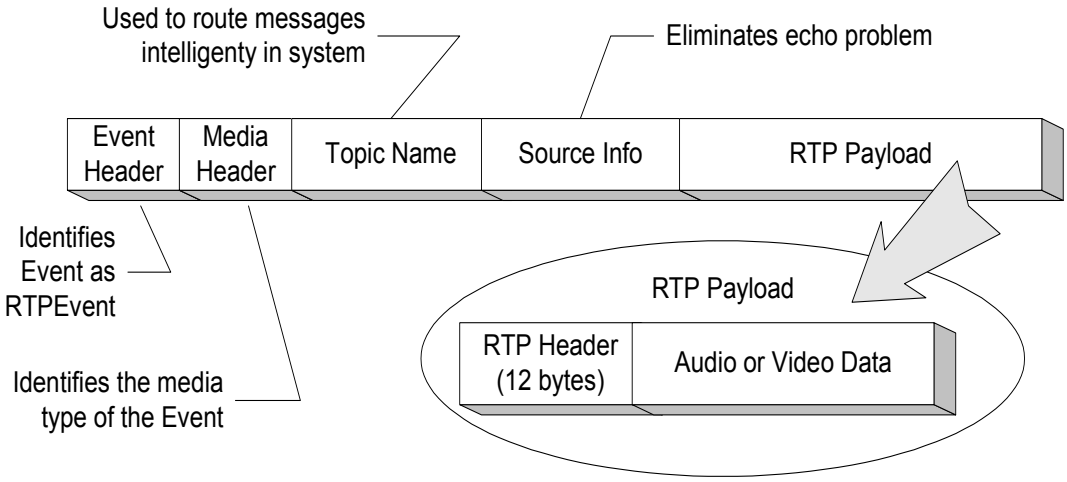

Figure 4-3 Serialized RTPEvent

We should also note the fact that when an RTPEvent package is traveling through multiple brokers, a 16 bytes dissemination trace value is added in the first broker and it is removed again in the last broker before sending it out to the destination client. Although, this adds extra load on the links among brokers, it should not affect the quality of the communications seriously, since these links are supposed to be high bandwidth. On the other hand, the links between clients and brokers will not have this extra load. These links are usually the most vulnerable links in the communication path

from sources to destinations. Therefore, the addition of the dissemination trace value should be tolerable.

4.4.5 Adding Support for Legacy RTP Clients

Although, we have developed a client application which can join a videoconferencing session and send/receive audio/video streams using RTPEvent messages, there are a lot of other RTP based clients that need to be supported such as VIC, RAT, Polycom systems, etc. These clients exchange RTP packages and use UDP or Multicast as the transport mechanism. Therefore, we have developed a specialized implementation of the NaradaBrokering transport framework [NBTRANSPORT] called RTPLink for UDP and MulticastRTPLink for multicast. This process entailed an implementation of the Link interface which abstracts the communication link between two entities. The RTP links can receive raw RTP packages over UDP or Multicast from legacy systems, wrap them in RTPEvents and propagate through the protocol layer in the broker node. Once it reaches the protocol layer, the event is routed within the distributed broker network.

An RTP media stream is composed of two different kinds of packages: RTP and RTCP packages. RTP packages carry the audio or video data along with the RTP headers which are used to provide a host of services such as payload type identification, sequence numbering, timestamp, source identification, etc. RTCP packages carry the control messages to monitor the timely delivery of real-time data. RTP and RTCP packages are exchanged on different ports. RTCP packages are exchanged on the port number following the RTP port number. Therefore, an RTP link implementation starts two sockets on these two ports. In addition, it publishes the RTP and RTCP messages

on different topic numbers. Similar to RTP protocol, RTCP packages are published on the topic number following the RTP topic number. Therefore, a pair of topic numbers is used to publish an RTP stream.

The RTPLink deals with the initialization, registration and data processing on the communication link. During the initialization process, RTPLink is provided a port number to listen for packages from the legacy client at the other end, and also the IP address and the port number of the legacy client to be able to send packages. For registration purposes, the RTPLink is assigned a NaradaBrokering-ID and it subscribes to the corresponding topic for its meeting. In the data processing part, the RTPLink constructs the RTPEvents for processing within the broker network when it receives media packages. On the other hand, when an RTPEvent is ready to be sent to the legacy application, RTPLink retrieves the RTP payload from the RTPEvent and sends it to the legacy application based on the parameters specified during initializations.

Since unicast and multicast RTP sessions are quite different, the link implementations for these two are also different. On unicast sessions, usually a user is in direct communication with another user and only one audio/video stream is exchanged through one link. On the other hand, in multicast sessions, there tends to be many audio or video streams in a session from many participants. Therefore, while unicast RTP link implementation is handling the communication with one user, multicast RTP link implementation should handle the communication with a group of users. MulticastRTPLink has two options: Either it can publish all streams to the same topic pair or it can publish each RTP stream on a different topic pair. When all streams are published on the same topic pair, all participants who subscribe to this topic receive

all the streams. Namely, they do not have any choice to select any specific stream in the group. On the other hand, if every stream is published on a different topic, then each user can select the stream of their choice. In practice, usually it makes sense to publish all audio streams on the same topic, since a user usually wants to hear all speakers in a session. However, it might be better for video streams to be published on different topics, since some users might only want to receive some video streams in a session.

A MulticastRTPLink examines the headers of the received RTP and RTCP packages and determines the packages that belong to the same RTP stream. Each RTP stream in a real-time session has a unique SSRC number. Then, it can publish all packages belonging to the same stream to the same topic number. Since many streams require many topics, the MulticastRTPLink should either be given a list of topic numbers to use when it is started, or it should ask for a new pair of topics whenever it receives a new RTP stream. It can also examine the content of RTCP packages to garbage collect the RTP streams when the session is ended by sending a bye message. In addition, it can generate events to mark the arrival and departure of RTP streams.

RTP links can be managed either statically or dynamically. By using a configuration file, all RTP links can be initialized when a broker is started. This is a very simple solution, but it is not very flexible. It requires the broker to be restarted to add/remove links. Dynamically managing the links requires having an RTPLinkManager running inside the broker, which listens for request to start/stop RTP links. It can listen on a JMS topic for those request messages. The application that sends these requests can either be running independently or it can be accessed through a web page. An administrator can add and remove RTP links at various brokers. In

addition, it is also possible to add/remove links programmatically without human intervention by implementing some system components that manage user joins and leaves automatically.

In the next chapter, we examine the performance of NaradaBrokering broker network for real-time audio and video delivery in both single broker and distributed settings.

Chapter 5

Performance Tests

In this chapter, we evaluate the performance of NaradaBrokering broker network for real-time audio and video distribution and determine the capacity of this messaging middleware. First of all, we identify the factors that impact the performance of a broker and analyze them in the context of real-time audio and video delivery. Secondly, we test the performance of a single broker extensively. Since the building blocks of the distributed broker network are brokers, it is essential to know thoroughly the capacity and the limits of a single broker. Moreover, we test distributed brokers to investigate the scalability and the performance of the broker network in distributed settings. Finally, we conduct performance tests on wide area networks.

5.1 Performance Analysis of a Broker

Before performing the audio/video tests, we investigate the factors that affect the performance and the scalability of a broker in the context of audio/video delivery. We identified four main factors to examine. These are:

1. The size of the audio and video packages.
2. The frequency of audio/video packages in a stream.
3. The number of subscribers on a topic or the number of outgoing streams from a broker.
4. The number of incoming streams to a broker.

We investigate the effects of these factors one by one. We conducted all performance tests in this section in a Linux cluster which had 8 identical machines with a gigabit network switch among them. We always ran the broker in a dedicated machine to measure the performance of the broker accurately. Direct gigabit network connection between any two machines in this cluster without connecting to any router, minimized network overheads. These machines had Double Intel Xeon 2.4GHz CPUs, 2GB of memory with Linux 2.4.22 kernel. Both NaradaBrokering software and our applications are developed in Java. Therefore, all components of the tests were running as Java applications by using JDK 1.4.2.

5.1.1 The Effects of Package Sizes

Audio/video streams are composed of relatively small size packages. Their sizes are usually less than 1 KB. There are two main reasons for this. First, the available audio/video data can be less than 1 KB at the time of encoding. Since these streams are real time, packages are transmitted periodically. When all the available data is encoded, often it is less than 1 KB. This is particularly true for audio streams. Many audio codecs transmit data every 20-90ms, and usually they produce packages that are less than 1 KB. The second reason is that codecs divide large size encoded data into smaller packages to utilize the underlying network protocols more efficiently. On Internet, the

links between any two parties have a maximum package size. It is called Maximum Transmission Unit (MTU). It is a property of IP layer. All packages higher than this value are fragmented at the IP layer. However, this can be more costly than fragmenting at the application layer. When even one fragment of a package is lost along the way, the whole package is assumed lost. On the other hand, if the fragmentation is done on application layer by a codec, each package is encoded to be meaningful by itself. Therefore, the effects of package losses can be minimized. This is recommended by [RFC2736] for RTP stream delivery over Internet. Although there is no universal value for MTU, most of them are below 1500 Bytes [RFC1191]. Many codecs set the maximum package size as 1KB. Here we investigate the effects of package sizes up to 2KB.

We investigated the effects of package sizes by setting up a meeting on a broker. Since it is difficult to measure and observe the effects with few packages and few subscribers, we set up a meeting with 500 receivers. A publisher transmitted a stream of fixed size packages to the meeting topic on the broker. It published 1000 packages with 50ms intervals for 50 seconds. These packages are delivered to all 500 subscribers by the broker. We gathered the results from the last subscriber to whom packages are always delivered last. Since the processing times on the sender and the receiver clients are very small and the transmission durations from the sender to the broker and from the broker to the receiver is also very small, almost all latency is introduced by the broker to route these packages to subscribers. Table 5-1 shows the results of these tests, and Figure 5-1 shows the graph of these results. The latency values at the second column show the average latencies of the 1000 packages delivered

to the last user in the meeting. The line in the graph shows the estimated linear function calculated on the basis of the experimental data.

Table 5-1 Package size test results

Package size (bytes)	Latency (ms)	Number of Receivers	Incoming Bandwidth (kbps)	Outgoing Bandwidth (Mbps)
100	9.37	500	16	8
200	9.87	500	32	16
400	9.93	500	64	32
600	10.46	500	96	48
800	10.54	500	128	64
1000	11.23	500	160	80
1200	11.62	500	192	96
1600	13.24	500	256	128
2000	13.87	500	320	160

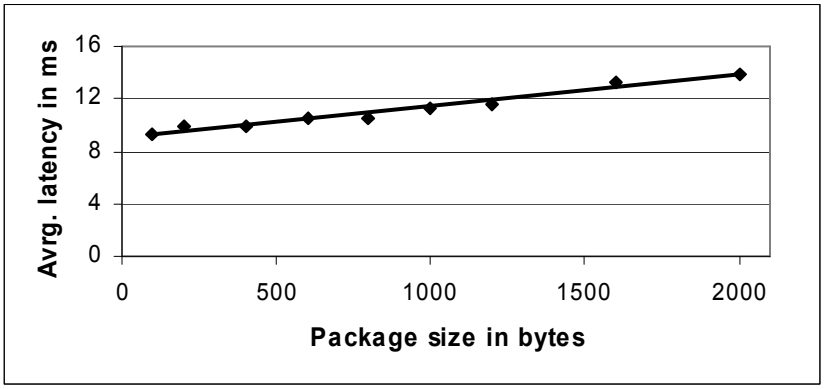

Figure 5-1 Package size test results

As we can see from Figure 5-1, the latency increases linearly with the increase on package sizes. Nonetheless, this increase is very limited. When we analyze this data by using the least square fitting method in MS Excel, we find the following linear function:

Latency(ps) = 9.04 + 0.0024*ps (Eq. 1)

where ps => package size in bytes.

The constant 9.04ms shows the minimum latency introduced by routing packages to 500 subscribers independent of their sizes. The second constant 0.0024 shows the slope of the linear function that indicates the latency introduced by one byte of the package size. If we calculate this slope for 100 bytes by multiplying with 100, it becomes 0.24ms. This means, an increase of 100 bytes in package size, results in an increase of 0.24 ms of latency when routing a package to 500 clients. For one client, it results in an increase of 0.00048 ms.

The most important result of this test is that the increase on package size does not result in an equal proportionate increase on latency. While there is a ten fold difference between a 1000 bytes package and a 100 bytes package, the latency introduced for the 1000 bytes package is only %23 higher than the latency of 100 bytes package. Therefore, it would be much better to send larger size packages than smaller ones as long as the broker is concerned. Another important observation is that since the delay is dominated by the number of packages rather than their sizes, we should pay more attention to the number of packages in a stream than its bandwidth.

5.1.2 The Effects of the Frequency of Audio/Video Packages

Both audio and video codecs encode the available data periodically and send out. Usually audio codecs send one package per processing interval. Video codecs may send multiple packages depending on the algorithms and the changes on video frames. When the frequency is higher (smaller package sending intervals), codecs usually provide higher quality streams. It is important to understand the impact of this frequency on the distribution network to use the available resources optimally.

We tested single audio meetings with two different package encoding intervals. We used ULAW audio codec in both cases. In one test, it encoded every 30ms and the size of each package was 240 bytes. In the other case, it encoded every 60ms and the size of the audio package was 480 bytes. There was only one audio meeting on the broker in each case. Table 5-2 and Figure 5-2 show the results. Both results are very similar until 1600 receivers. Since the 60ms interval case has larger package sizes, its average latency is a little higher than the 30ms interval case, but not significantly different. When there are 1600 receivers in the 30 ms interval case, the broker is saturated and it is unable to process packages on time. The routing of a package on the broker takes more time than the arrival interval of the packages (30ms), and each package is delayed by the previous ones. Therefore, it results in a steady increase on the latency values. Since the package sending interval is much higher on the 60 ms case, it can support up to 2800 receivers in a meeting. The broker does not become saturated unless it takes 60ms to deliver a package to all its subscribers. While the broker supports 2800 participants in an audio meeting when the package sending interval is 60ms, it supports only 1500 participants when the package sending interval is 30ms.

Therefore, the package sending interval affects the scalability of a broker significantly. The higher the frequency of an audio stream is the more loads it puts on the broker.

Table 5-2 Audio meeting tests for different package sending intervals

Number of Receivers	Avrg. latency 30 ms intervals	Avrg. latency 60 ms intervals
200	2.3	2.65
400	4.2	4.75
800	8	9.35
1200	11.6	13.8
1500	17.8	NA
1600	2275	15.4
2000	NA	21.4
2400	NA	24.75
2800	NA	35.03
3000	NA	2742

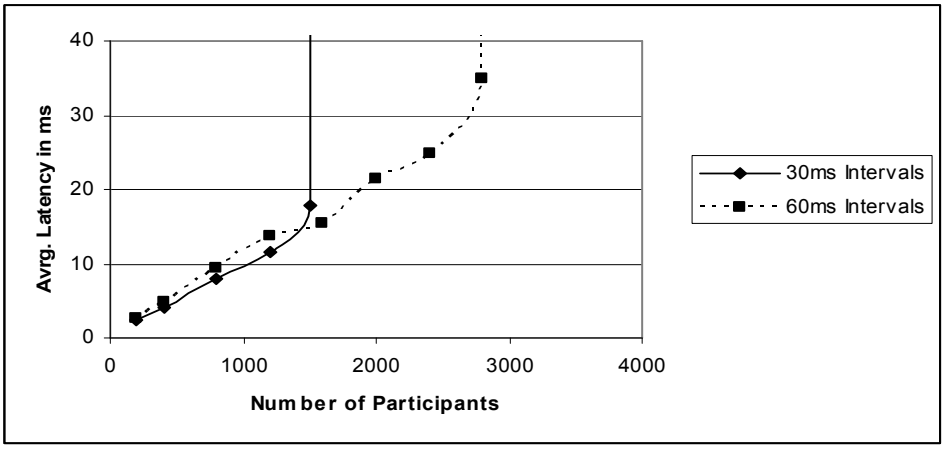

Figure 5-2 Test results for audio package sending frequency

In the case of video streams, the important parameter is the frames per second (fps). This shows the number of frames to be encoded per second. If it is 10, the video codec encodes 10 frames per second, one frame every 100ms. If it is 20, it encodes one frame every 50ms. When frames are encoded more often, they produce more packages and they require more bandwidth. However, the number of packages does not increase equally with the increase on fps. When we double the fps, the number of transmitted packages is not doubled. Because, when frames are encoded more often, there is less change on the picture between consecutive frames. This reduces the number of packages to be generated. For example, we counted the number of packages in an H.261 stream for 2 minutes. When the fps was 10, it generated 2589 packages. When the fps was 20, it generated 4027 packages. The number of generated packages increased only 1.55 times, when the fps is doubled. Therefore, we can say that the frequency of video encoding affects the performance of the broker significantly. However, its impact is less than the effects of the frequency of audio encoding.

5.1.3 The Effects of the Number of Outgoing Streams

When the number of participants in a meeting increases, the load on the broker also increases. When a package arrives at the broker, if there is no other package either being routed or waiting in the queue, the broker starts routing this package immediately. It routes the package to subscribers in first-come-first-serve basis. It first routes to the client who subscribed first, then it routes to the second subscriber, and so on. Therefore, while the first subscriber always gets the best service, the last subscriber always gets the worst service. Although this approach might seem to be favoring the

initial subscribers over the later ones, it provides services with minimum jitter for all subscribers.

Here we investigate the load introduced by the number of subscribers on a meeting and its effect on the latency experienced by the receivers. We define the latency (L) as the total time it takes for a package to reach to its destination from the sender:

$$\mathbf{L = RT - ST} \qquad \text{(Eq. 2)}$$

RT = Package receive time at the receiver.

ST = Package sent time from the sender.

There are a number of factors that contribute to the latency. We can formulate them as follows:

$$\mathbf{L(n) = TT1 + WT + RT(n) + TT2} \qquad \text{(Eq. 3)}$$

L(n): the latency for the nth subscriber on a topic.

TT1: the transmission time from sender to the broker.

WT: the waiting time before the broker starts routing a package.

RT(n): the routing time it takes for the broker to route the nth package.

TT2: the transmission time from broker to the receiver.

We can also formulate the routing time as follows:

$$\mathbf{RT(n) = PT + ST*n} \qquad \text{(Eq. 4)}$$

PT: The processing time it takes for a broker to pick up a package from the queue and calculate the destinations in the broker. This can be considered constant as long as the number of subscribers is not in millions. In our tests, since we have at most a few thousand subscribers, we consider it as a constant value.

ST: The amount of time it takes to send a package. This can also be considered constant as long as the broker is running on the same machine.

We investigate the effects of the number of outgoing streams or the number of participants in a meeting by running audio meetings on a broker with different number of participants. A client transmitted an audio stream into the meeting for every test. Since the audio codec sends periodic equal size packages, it makes easier to evaluate the results and identify the effects of the number of outgoing streams. The transmitter client published a ULAW audio stream with 64kbps bandwidth. It sent a package every 30ms with 240 bytes. We gathered the results from the last client in the meetings. Table 5-3 shows the results and Figure 5-3 shows the graph of these tests.

Table 5-3 Latencies of the last user in single audio meeting tests

Number of receivers	Latency (ms)
200	4.11
400	8.13
600	11.46
800	15.27
1000	19.4
1200	22.46

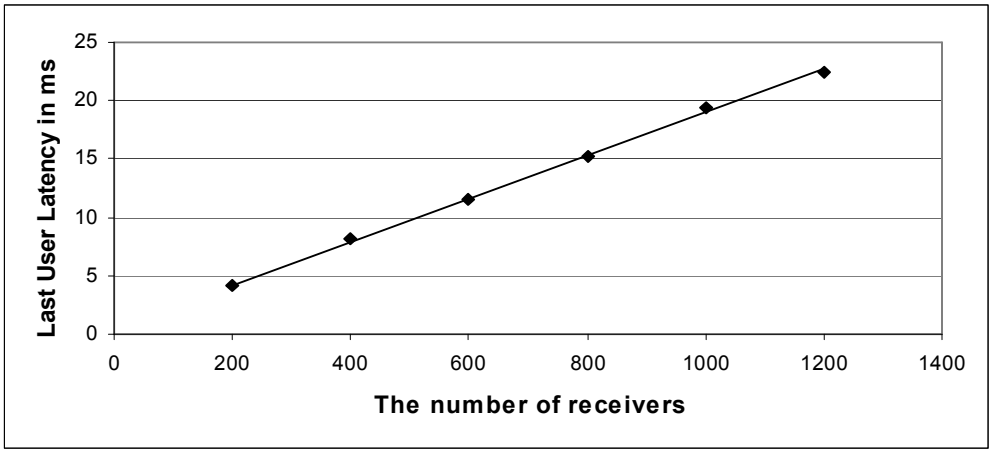

Figure 5-3 Latency of the last user in single audio meeting tests

For this test, the waiting time in the (Eq. 3) is zero. Because, there is only one meeting and the routing of each package is completed before the next one arrives. Then, the equation becomes:

L(n) = TT1 + PT + ST*n + TT2 (Eq. 5)

Since TT1, TT2 and PT are constant for all tests, we can represent them as one constant.

c1 = TT1 + PT + TT2 (Eq. 6)

L(n) = c1 + ST*n (Eq. 7)

When we calculate the linear function of Table 5-3 with the least square fitting method in MS Excel, we get the following equation:

L(n) = 0.53 + 0.0185*n (Eq. 8)

In this equation, 0.0185 represents the sending time (ST). Namely, it takes 0.0185ms for the broker to send a package in this setting to a client. Therefore, every participant, who subscribed earlier, adds up 0.0185ms of latency to the routing times of

the later subscribers. 0.53ms represents the total delay added by the transmission and processing times. In addition, (Eq. 8) helps us to estimate the amount of time it takes for the broker to route a package to n subscribers on a topic. For example, we can calculate that it takes 1.85ms for the broker to send a package to 100 participants and 18.5ms to send to 1000 participants. This is very helpful both when estimating the load on the broker and the latency introduced by the broker.

5.1.4 The Effects of the Number of Incoming Streams

It is also important to understand the effects of having multiple concurrent meetings on a broker. We investigated this by having multiple same size audio meetings at the broker. There was an audio publisher for each meeting. The transmitting times of these publishers are very important to determine the waiting times of packages at the broker. In the worst case scenario, all publishers might publish almost at the same time and the last arriving package needs to wait all others to be routed. In the best case scenario, all publishers might publish evenly distributed on time by causing minimum waiting delays to one another. However, on real life scenarios it is more likely for the publishers to be independent of one another and to be randomly distributed on time. Therefore, we first investigate the worst case scenario and then the random distribution of publishers.

We tested the worst case scenario by having multiple audio meetings on a single broker. One audio publisher sent the same audio stream to all meetings. It sent each audio package to all meetings one after another without any delay. Therefore, the audio packages for the later meetings wait the earlier ones to be routed at the broker. We

gather the results from the last receiver in the last meeting. We can formulate the latency for this receiver as follows:

$$\mathbf{L(m, n) = TT1 + WT(m, n) + RT(n) + TT2} \quad \text{(Eq. 9)}$$

n: The size of meetings. All meetings are the same size.

m: The total number of meetings.

Since the audio package of the last meeting needs to wait on the broker for all other packages to be routed, the waiting time is:

$$\mathbf{WT(m, n) = RT(n)*(m-1)} \quad \text{(Eq. 10)}$$

When we put this into (Eq. 9):

$$\mathbf{L(m, n) = TT1 + RT(n)*(m-1)+ RT(n) + TT2} \quad \text{(Eq. 11)}$$

$$\mathbf{L(m, n) = TT1 + RT(n)*m+ TT2} \quad \text{(Eq. 12)}$$

We can calculate the routing time of a package for a meeting size of n as:

$$\mathbf{RT(n) = PT + ST*n} \quad \text{(Eq 13)}$$

When we put this in (Eq. 12),

$$\mathbf{L(m, n) = TT1 + (PT + ST*n)*m+ TT2} \quad \text{(Eq. 14)}$$

$$\mathbf{L(m, n) = TT1+ TT2 + PT*m + ST*m*n} \quad \text{(Eq. 15)}$$

Table 5-4 shows the experimental results of the latency values for the last user in the last audio meeting for multiple concurrent audio meetings with the same audio stream as the previous single meeting tests.

When we calculate the constants of Eq 15 based on the experimental results of Table 5-4 using the least square fitting method in MS Excel, we get the following equation:

$$\mathbf{L(m, n) = 0.41 + 0.045*m +0.0176*m*n} \quad \text{(Eq. 16)}$$

Table 5-4 Latencies of the last user in the last meeting in multiple audio meetings.

Latency (ms)	Number of Meetings	All receivers	Meeting size
9.49	40	400	10
11.53	50	500	10
13.66	60	600	10
15.57	70	700	10
8.36	20	400	20
12.28	30	600	20
16.51	40	800	20
20.15	50	1000	20
7.61	10	400	40
11.61	15	600	40
15.35	20	800	40
19.15	25	1000	40

The values at this equation show the values of the constants at (Eq.15). 0.0176ms represents the sending time (ST) of a package. It is very close to the previous result from the single meeting tests. The difference is negligible and it can be explained by the errors introduced from the calculations and from the experimental setups. 0.045ms represents the processing time (PT). Therefore, the processing time is 2.5 times higher than the sending time. Namely, it is 2.5 times more costly to receive and process a package than sending it to subscribers. 0.41ms shows the total latency introduced by the transmissions from the sender to the broker and from the broker to the receiver.

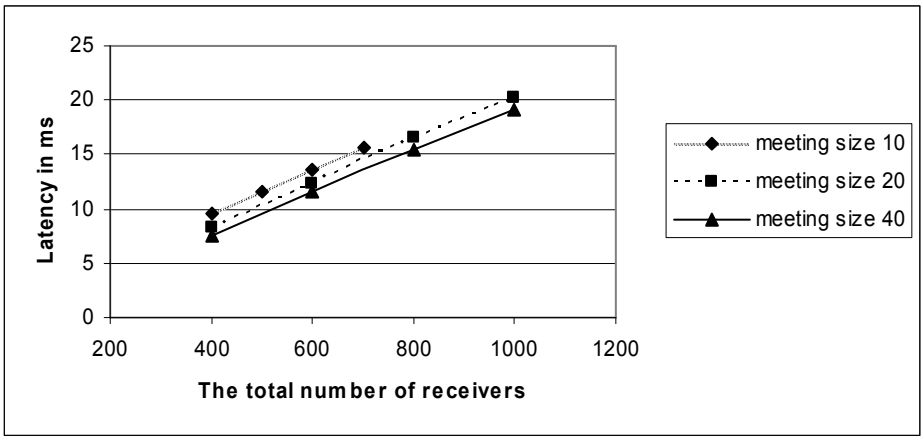

Figure 5-4 Latencies of multiple audio meetings for various meeting sizes

Figure 5-4 shows the graph of the data on Table 5-4 according to the meeting sizes. As it can be seen, when the meeting size gets smaller, the latency increases for the same number of total receivers in the broker. The latest receiver of the meeting size 10 has the highest latency and the latest receiver of the meeting size 40 has the smallest latency. The reason for this is the higher number of incoming streams for smaller size meetings for the same number of participants in total. In summary, this test shows that when the number of incoming audio streams to a broker increases, the total number of supported participants decreases. However, as we will show shortly, the quality of service tends to be much better when there are multiple concurrent meetings on a broker because of the better utilization of the broker resources.

Now we investigate the effects of having independent publishers for each audio meeting. We performed multiple concurrent audio meeting tests with independent publishers, each publishing their streams randomly distributed on time. We conducted many tests with different number of meetings. The number of participants in each

meeting was 20. Since there is no worst or best meeting in the case of independent transmitters, we measured the latencies from 10 meetings. Then, we calculated the average latencies of these 10 users. We compare the results of these tests with the results of single audio meeting tests and with the results of the worst case of multiple audio meetings. To be able to compare the results, we computed the average latencies for those tests, too. In single meeting tests, we averaged the results from the best, the middle and the worst clients for each meeting. In the worst case of the multiple concurrent meetings, we averaged the results from the best, the middle and the worst meetings. The meeting sizes for the worst case of multiple concurrent meetings was also 20 to make the comparison easier. Figure 5-5 shows the results of these tests on a graph. It shows the average latency values for the total number of participants on the broker.

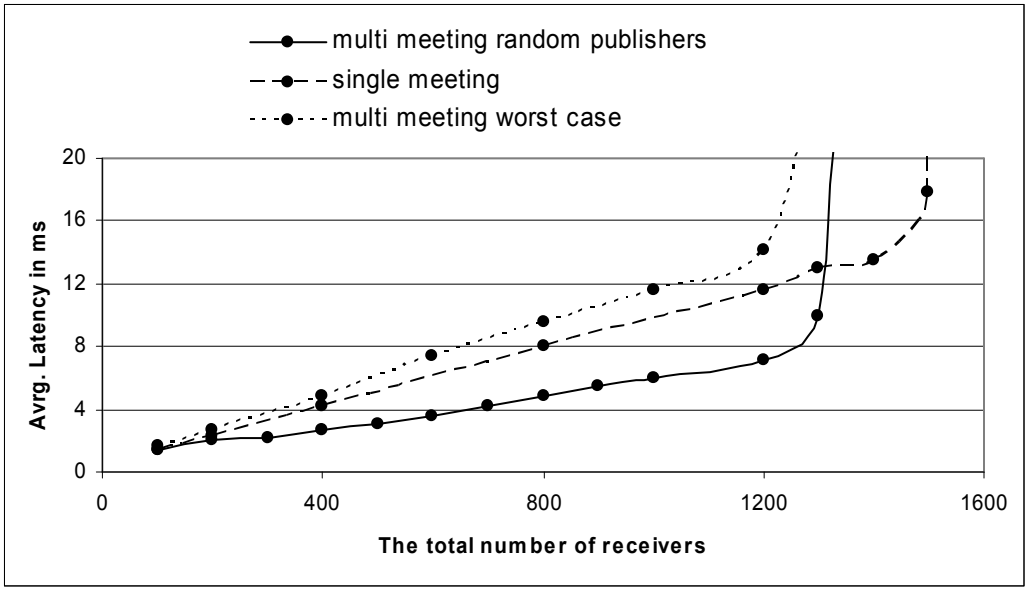

Figure 5-5 Performance comparisons of audio meetings

As it can be seen from Figure 5-5, the random publishers' case provides much smaller latency values than others until the broker is overloaded. The main reason for this is the better utilization of the broker. Since audio packages arrive on the broker with random distribution on a given sending interval, packages tend to wait much less time compared to worst case scenario in which almost all packages arrive at the same time. It is also much better than the single meeting case. Although in single meeting case there is no waiting for other meetings and the broker starts serving the received packages immediately, the participants need to wait for other receivers on the same meeting. Since the meeting size of the single meeting is much higher, the later subscribers need to wait the earlier ones in the queue before being served. This increases the average latencies of the single meeting participants. Therefore, the broker provides much lower latency values on the average for the same number of participants when there are multiple audio meetings. This test also shows that the broker gets overloaded earlier in multi meeting cases than the single meeting case. This is because of the overhead of the processing of incoming streams on the broker. This overhead is not apparent when the broker is not fully loaded, because it is distributed over time.

We can summarize the results of analyzing the performance of a broker as follows.

- The effects of the sizes of audio and video packages on latency are very small when routing through the NB network. The important factor is the number of packages being routed, not the size of each package.

- When transferring the same audio or video stream, the package sending frequency affects the performance and the scalability significantly. The higher the frequency is the more loads it puts on the broker.
- The number of subscribers or the number of outgoing streams in a meeting affects the latency linearly. It takes a constant amount of time to support each receiver.
- The broker provides better quality of service when there are multiple concurrent meetings compared to having a large size meeting. The broker is utilized better in the multi meeting cases and the average latencies provided to receivers are much smaller.

After investigating the factors that impact the latency of packages and the performance of the broker, now we can proceed with the audio and video tests. First, we explain the audio and video streams that we used for the tests and we outline the criteria for the evaluation of the quality of audio and video communications.

5.2 The Characteristics of Audio and Video Streams

We need to know the characteristics of audio and video streams thoroughly to be able to test the scalability and the performance of the broker network accurately, and to be able to understand and evaluate the results properly. The characteristics of audio and video streams are significantly different, though with some similarities. First of all, video streams tend to be much more bandwidth intensive. Secondly, most audio codecs send periodic packages with fixed size data during a session. The only exception occurs when there is a silence period in audio. In that case, no audio packages are sent if the encoder is suppressing the silence. Similarly, video codecs also encode each frame of

the stream periodically according to the frame rate of the video stream. Nonetheless, they may generate multiple packages with differing sizes for each video frame according to the changes in the picture between consecutive frames. If there are more changes, more packages are generated for a frame. If there is less action, fewer packages are generated. Moreover, video codecs occasionally may send full picture updates. These full pictures generate much more packages than regular frame encodings in which only the changes are encoded from the previous frame. Consequently, neither the size of the video packages nor the number of video packages per frame is constant. Therefore, we have tested the performance and the scalability of the brokering network in three different settings: audio only meetings, video only meetings, and audio and video combined meetings.

There are many types of audio and video codecs to be used when performing these tests. We have chosen widely used codecs with average bandwidth requirements. We have used the same audio and video streams in all tests to be able to compare multiple test results. We have recorded an audio stream and a video stream for 2 minutes to use in these tests.

We chose ULAW [G711] as the audio format. The bandwidth of the recorded audio stream was 64 kbps. The audio codec sent an RTP package every 30 millisecond. All audio packages were 252 bytes long. There were 4100 audio packages in total. There was no silence period, so packages are continually transmitted during the session. This is a telephone quality audio stream and widely used in videoconferencing sessions over Internet.

We chose H.263 [H263] as the format of the video stream. The recorded video stream had the average bandwidth of 280 kbps. The frame rate of the video stream was 15 frames per second. It was the recorded video stream of a speaking participant in a video conferencing session. Although the average bandwidth was 280 kbps, the bandwidth was fluctuating mostly between 250 kbps to 310 kbps throughout the recording. The transmitter transmitted 1800 frames during 2 minutes, which had 5610 packages in total. The video codec was dividing the frames that have more than 1 KB of data into multiple packages. The average length of the video packages was 740 bytes. The video codec sent one full picture update every 60 frames or every 4 seconds. These frames had much more packages than regular frames as it can be seen from Figure 5-6. It shows the number of packages per frame. Although, a frame had 3.1 packages on the average, the number of packages for full updates fluctuated from 10 to 18. This video can also be considered an average video stream with good quality for a videoconferencing session.

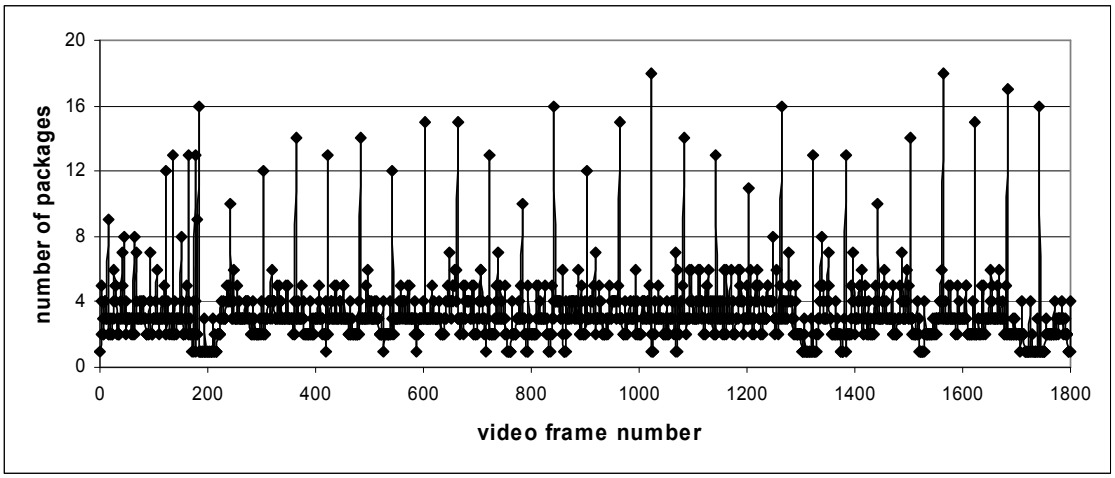

Figure 5-6 Number of packages per frame in the video stream

5.3 Quality Assessment of Media Delivery

There are three important factors that affect the quality of audio and video communication when transferring streams over Internet. These are the latency, the jitter and the loss rates of packages [CLAYPOOL]. The values of these factors provide valuable information to assess the quality of the audio and video delivery.

International Telecommunications Union [G114] recommends that the mouth-to-ear transit delay of one-way audio should not exceed 400ms to provide an acceptable quality conversation among the remote participants. It also points out that the delays up to 300ms satisfy almost all users by delivering good quality. Though, 150ms is preferred to provide excellent quality. On the other hand, there is no agreed upon standard for the transmission delay of video. Nonetheless, in most cases video is used along with audio and it should have a similar latency requirement to be in synchrony with audio. Therefore, we require the same latency values both for audio and video.

The total transit delay is the combination of a number of factors: the processing time on the sender and receiver machines, the transit time from the transmitter to the broker, the routing time at the broker, and the transit time from the broker to the receiver. In our tests, the transit times from the transmitter to the broker and from the broker to the receiver is almost zero, since they are all running in a gigabit subnet. We also ignore the processing time on sender and receiver machines by recording the times of the packages on the transmitter machine after they are encoded and before the decoding on the receiver machine. Therefore, almost all the latency is introduced by the broker to route the packages. Since in real life applications the total latency should not exceed 400ms for acceptable quality and 300ms for good quality, we require that a

broker should not add more than 100ms of latency. This allows the remaining two transit delays and the processing on receiver and transmitter machines to introduce up to 200ms for good quality and up to 300ms for acceptable quality. We label those packages with higher latency values than 100ms as late packages. Although these packages are delivered to destinations, they might be dropped without being played, since they do not arrive on time. Therefore, we consider them as lost packages and assess the quality of the transmission accordingly.

Jitter [RTP] is defined as the variation in the arrival times of the packages at the receiving end. Since Internet neither provides guaranteed package delivery nor constant delay, packages may take different times to get to the destinations. To smooth out these variations in the arrival of packages, receiving ends implement a playout buffer algorithm [RAMJEE]. These algorithms delay the playing of packages that arrive earlier to be able to compensate the late ones. However, they discard packages that arrive too late without playing to the user. Although there is no formal recommendation about the jitter, [CALYAM] recommends that the jitter values below 20 ms provide good quality. The jitter values between 20- 50 ms provide acceptable quality and the jitter values beyond 50 ms, results in poor quality. Therefore, in our tests, we require the broker network to introduce less than 10ms of jitter, so that the other transmission links can introduce more.

There are two factors that contribute to the package losses experienced by the receiving end. First one is the losses during the transmission. Second one is the losses due to the late arriving packages. In our tests, there were no package losses because of the transmission. It is suggested in [COTTRELL] that the loss rates should be less than

1.0% for good quality, 1-2.5% for acceptable quality, 2.5-5.0% for poor quality, and 5.0-12.0% for very poor quality. Therefore, in our tests, we tolerate only around 1.0% of package losses.

5.4 Performance Tests for One Broker

We conducted extensive tests to evaluate the performance and the limits of a single NaradaBrokering broker. Since the building blocks of the distributed brokering network are brokers, it is essential to know thoroughly the capacity and the limits of a single broker. Knowing the capacity and the performance of a single broker helps us to predict the performance of the broker network in distributed settings. In addition, it helps us to identify the bottlenecks and problems in multi broker environments. We tested two cases thoroughly. The first one is single large scale meetings. The second one is multiple small scale meetings.

In our tests, each meeting is designed as a single speaker meeting. There is one speaker and many listeners in every meeting. The speaker client transmits an audio or a video stream to the meeting and listener clients receive that stream. Each meeting has a dedicated topic to which the speaker publishes the audio or the video stream. Listeners subscribe to the meeting topic to receive the media stream through the broker. When there are multiple meetings, each meeting has its own topic, and its own speaker and listener clients. In the case of audio and video combined meetings, audio and video meetings are completely independent. Both of them are seen as separate meetings by the broker network.

The multiple speaker meetings are equivalent to multiple single speaker meetings as long as the delivery of streams is concerned by the broker network. Having a

meeting of N participants with M speakers are equal to having M single speaker meetings each having N participants. In both cases, M streams are delivered to N clients. Therefore, we have not tested multiple speaker meetings.

We tried to limit the outside factors that might introduce extra overhead in these tests. First of all, we tried to avoid interferences by other processes in the machines which we use for tests. We dedicated one machine entirely for running the broker. We also dedicated another machine to run the clients that send and receive media streams from which we gathered the results. In addition, we tried to minimize the affects of the network overhead by running the broker, transmitters and receivers in the Linux cluster that had direct gigabit network connection between the nodes. This cluster was the same Linux cluster that we used for the tests on the performance analysis of a broker, section 5.1.

5.4.1 Single Meeting Tests

We tested the performance and the scalability of a single broker for three types of single meetings: single audio meetings, single video meetings, and audio and video combined meetings. We started from small size meetings and went up to large size meetings until the broker can not deliver an acceptable performance. Since the characteristics of audio and video streams are significantly different, it is necessary to test each of these cases independently.

We have used Java Media Framework (JMF) [JMF] library to develop the transmitter and receiver clients. The transmitter client read a media file from the hard drive and published the audio or video packages on the broker by encoding them. It also recorded the sending times of all media packages. There were two types of

receiver clients. Measuring receivers were receiving the stream and recording the arrival times of packages. Passive receivers were receiving the stream from the meeting and simply discarding them. We used the following setting in Figure 5-7 to perform the single meeting tests. For each meeting, the transmitter and measuring receivers were running in the same machine to avoid the clock synchronization problems when calculating the latencies. Other passive receivers were distributed among the remaining 6 machines in the Linux cluster. We calculated the latencies for each of the 12 measuring receivers. We simply subtracted the sent times from the received times by using millisecond resolution. Then we averaged these 12 latencies to get a better idea about the overall characteristics of the stream delivery. When calculating the latencies, we ignored the first 100 packages to compensate for start up costs.

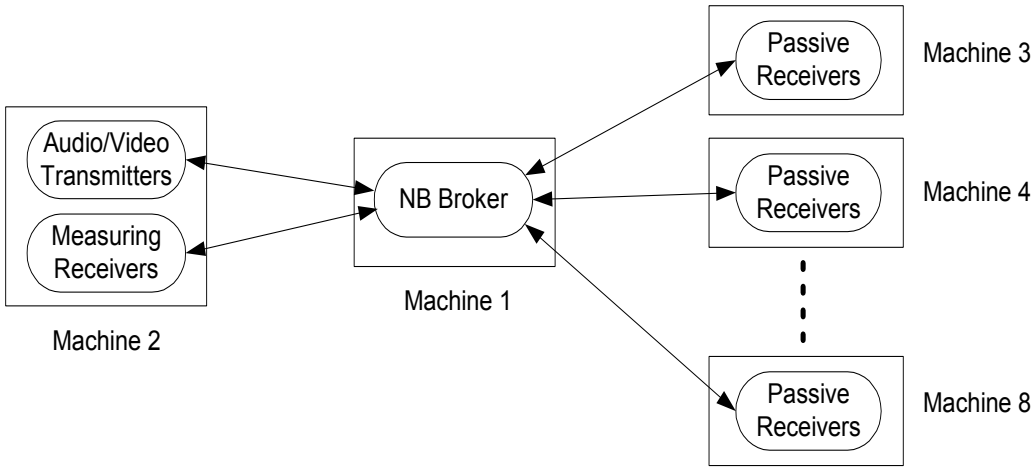

Figure 5-7 Single meeting test setting for one broker

JMF library implements the RTCP specification in RTP protocol to generate RTCP packages during the audio and video communications. Although, in online meetings the receiving participants do not send any RTP packages, they send RTCP

packages to report to other participants and to the transmitter client about their presence in the meeting and the quality of service they are receiving. In small size meetings, the extra load of RTCP packages is negligible. But in large scale meetings, the load of RTCP packages can become a bottleneck, since each RTCP package is delivered to every other participant in the meeting. To avoid this, RTP specification [RTP] requires scaling down the sending interval of RTCP packages when the number of participants in a session increases. It recommends that RTCP packages should use %5 of the total session bandwidth. However, our tests showed that JMF library does not scale down the RTCP package sending intervals in large size meetings. The distribution of RTCP packages dominated the large size meetings. Therefore, we have disabled RTCP package delivery in our tests. However, this should not affect the performance tests of the broker significantly, since RTCP packages must use at most %5 of the session bandwidth.

5.4.1.1 Single Audio Meeting Tests

We tested single audio meetings with varying number of participants on a single broker. Table 5-5 shows the summary of the results of these tests. Each row in the table shows a test case with the given number of participants in the meeting at the first column. Second column shows the average latencies of the first subscribed user in the meeting topic. Third and fourth columns show the average latencies of the middle and last registered clients, respectively. Since the routing algorithm in the broker routes the packages in the first-come-first-serve fashion, clients are served in the order of their subscription with the meeting topic. The client, who subscribes first, is always served first. The client, who subscribed last, is always served last. Therefore the latency values

of the first client is almost always the same and it does not change by the number of participants in the meeting, until the broker is overloaded and starts delivering packages very late. We should remember that the audio packages are sent periodically (a package every 30ms) and the routing of a package does not affect the routing of the next one, until the broker is overloaded. The broker is overloaded when it can no longer finish delivering an audio package to all participants in the meeting before the next audio package arrives. In such a case, each audio package delays the delivery of the next one and the routing time of each package increases continuously. In this test, the broker is overloaded when there were 1600 participants in the meeting. Figure 5-8 shows the latency values of the middle user when there are 1600 participants in the meeting for 4000 packages transmitted. The latency values keep increasing for each upcoming package.

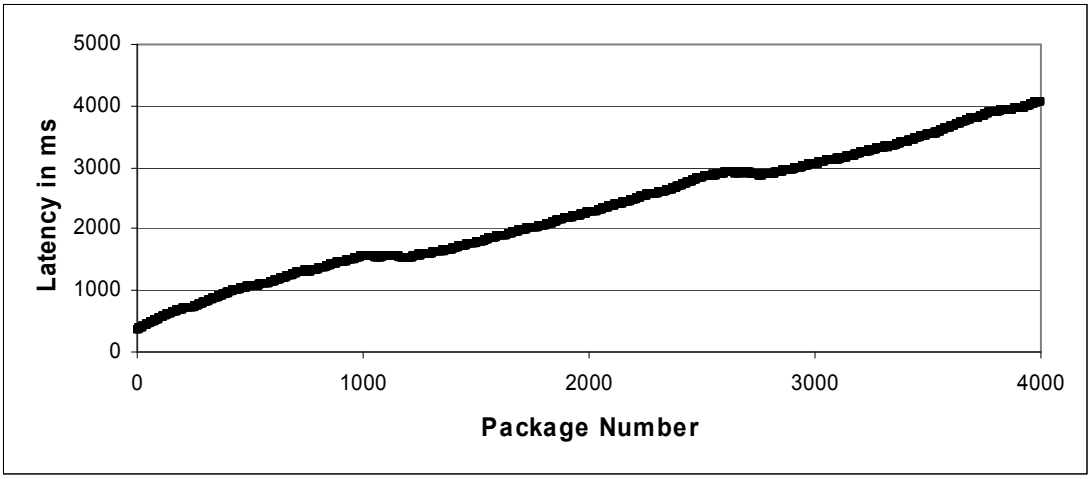

Figure 5-8 Latency values for the middle user in single audio meeting with 1600 participants

Fifth column shows the average latencies of 12 participants from which we gather the results. These are the first 4, middle 4, and last 4 clients, in the order of their

subscription to the meeting topic. Sixth column shows the average jitter values calculated for the average latencies of 12 participants. Jitter values are calculated according to the formula given in RTP specification [RTP]. Since the routing of consecutive packages does not affect each other, the jitter values are very small. Seventh column shows the percentages of packages that arrive later than 100ms. There are no late arriving packages until the broker is overloaded. Eighth column shows the amount of data the broker receives from the publisher. Last column shows the outgoing bandwidth from the broker, the amount of data sent out to the participants.

Table 5-5 Test results of single audio meetings for one broker

Number of Clients	First Latency (ms)	Middle Latency (ms)	Last Latency (ms)	Avrg. Latency (ms)	Avrg. Jitter (ms)	Avrg. Late Arrivals	In BW (kbps)	Out BW (Mbps)
12	0.5	0.7	0.7	0.6	0.18	0	64	0.76
100	0.5	1.4	2.3	1.4	0.15	0	64	6.4
200	0.5	2.3	4.1	2.3	0.18	0	64	12.8
400	0.5	4.2	7.9	4.2	0.21	0	64	25.6
800	0.5	7.9	15.5	8	0.18	0	64	51.2
1200	0.5	11.6	22.6	11.6	0.22	0	64	76.8
1400	0.5	13.5	26.5	13.5	0.26	0	64	89.6
1500	3.3	17.8	32.3	17.8	0.44	% 0.25	64	96.0
1600	2260	2275	2290	2275	1.2	% 100	64	102.4

These tests show that a broker can support up to 1500 participants in one audio meeting by providing very good quality audio with very few late arriving packages.

This result is consistent with the formula given at (Eq. 8) in the previous section. That equation suggests that the broker will be saturated when there are 1592 receivers. We should also remember that the scalability of the broker depends on the chosen audio stream and the capacity of the machine the broker is running on. If we had sent the same audio stream with 20ms intervals, the broker would have been overloaded around 1000 receivers. On the other hand, it would support much more participants if we had used another codec with higher package sending interval. Similarly, a higher capacity machine could support more participants and a lower capacity machine would support fewer participants.

5.4.1.2 Single Video Meeting Tests

Similar to single audio meetings, we tested the performance of a broker for single video meetings with varying number of participants. Table 5-6 shows the results of these tests. The columns of Table 5-6 are similar to that of Table 5-5. The only difference is the column 7 that has the results for late arriving packages of the last subscriber in meetings. Since there are more late arriving packages for video meeting tests, we included the late arrival rate of the last subscriber in the meting that gets the highest number of late arriving packages.

Contrary to single audio meeting tests, the delay and the jitter increases significantly even for the first participant as the number of participants grow. There are two reasons for this. The first reason is the multiple packages sent for each frame. All the packages in a frame need to wait the earlier ones to be routed on the broker except the first package in the frame. Therefore, even the first receiver experiences an increase on the latency and the jitter values as the number of participants grows in meetings. The

second reason is the uneven distribution of packages in the video stream. As the number of participants grows in meetings, the routing of packages for frames with higher number of packages can not be completed before the packages for the next frame arrives. In such a case, the packages for the upcoming frame wait the routing of the packages for the previous frame. This also results in extra latencies and jitters for the first participants in meetings.

The number of supported participants in a video meeting is much smaller than the number of supported participants in an audio meeting. Table 5-6 shows that 400 participants in this video meeting can be supported with very low latency and jitter, and very few late arriving packages. In the case of 500 participants, the number of late arriving packages increases to %3.0 for the last subscriber. The average jitter also goes up to 10 ms for the same test. Therefore the quality of the stream delivery for 500 participants becomes unacceptable, although the broker is overloaded when there are 1000 participants in the meeting. There are two reasons for this. First one is that the number of packages in the video stream is much higher than the number of packages transmitted in the audio stream. Another more important reason is the uneven distribution of packages in the video stream throughout the transmission. The number of packages transmitted for each frame changes according to the actions in the picture. Moreover, the full picture updates sent by the video codec have much more video packages than other regular frames as it can be seen at Figure 5-6. Since the video codec sends all packages in the same frame one after another without any delay, the later packages wait the earlier ones to be routed. Figure 5-9 shows the latency values of all 5610 packages transmitted for the last receiver in the video meeting with 400

receivers. The peaks on the latency graph correspond to the full picture updates. We can also estimate the latency of packages in this test using the equation (Eq.8). According to (Eq. 8), it takes 0.0185ms for the broker to send a package. It would take 7.4ms to route a package to 400 receivers. Therefore, it would take 22.9ms to route 3.1 packages for an average frame. However, it would take more than 100ms to route the full update frames that has 14 or more packages. The test results are consistent with these estimations as it can be seen from Figure 5-9. In summary, compared to single audio meetings, single video meetings utilize the broker resources poorly and the uneven distribution of packages in video streams results in late package arrivals long before the broker is overloaded. This limits the number of supported participants in single video meetings significantly.

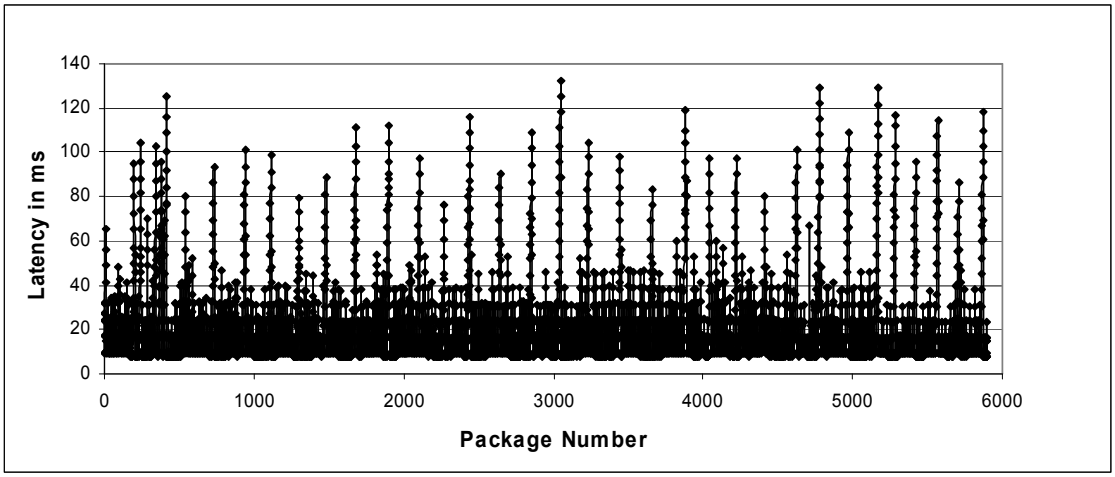

Figure 5-9 Latency values for the last receiver in single video meeting with 400 participants

Table 5-6 Test results of single video meetings for one broker

Number of Clients	First latency (ms)	Middle latency (ms)	Last latency (ms)	Avrg. latency (ms)	Avrg. Jitter (ms)	Last Late arrivals	Avrg. Late arrivals	In BW kbps	Out BW Mbps
12	1	1.2	1.3	1.2	0.44	0	0	280	3.3
100	3.1	4	5	4	2	0	0	280	27.8
200	6.3	8.3	10.2	8.3	4.7	0	0	280	55.6
300	10.2	13.2	16.2	13.2	7.8	0	0	280	83.4
400	13.4	17.3	21.2	17.3	10.1	% 0.75	% 0.05	280	111.2
500	18.2	23.4	28.5	23.4	13.2	% 3.0	% 2.6	280	139
600	22.6	28.6	34.5	28.6	15.5	% 5.1	% 4.3	280	166.8
700	29.8	36.8	43.7	36.8	18.1	% 8.4	% 7.6	280	194.6
800	45.6	53.6	61.6	53.6	21.3	% 17.6	% 16.2	280	222.4
900	93.7	102.7	111.7	102.7	23.8	% 40.8	% 38.2	280	250.2
1000	1599	1609	1619	1609	27.8	% 99	% 98.9	280	278

We should also note that the broker in single audio meeting tests gets overloaded when its outgoing bandwidth is 102 Mbps and it gets overloaded in single video meeting tests when its outgoing bandwidth is 278 Mbps. The reason for this big difference is the package sizes of audio and video streams. While the sizes of audio packages are 252 bytes each, the average video package size is 752 bytes. As we have shown previously, the dominant factor is not the bandwidth but rather the total number of packages in these streams.

5.4.1.3 Audio and Video Combined Meeting Tests

Contrary to the pervious two cases, in this test there are two concurrent meetings each having one media stream. The delivery of one stream affects the delivery of the other. Therefore, we can see the effect of one stream on another and vice versa.

When we performed some initial audio and video combined meeting tests, we observed that the delivery of video streams affected the delivery of audio streams significantly. There was only one queue at the broker for all packages and the routing algorithm routed the packages in the first-come-first-serve basis. Therefore, audio packages needed to wait for the video packages. Although, the audio packages are evenly distributed on time and the latency of consecutive packages for a user is the same in a meeting, when an audio meeting is held together with a video meeting, the latency graph of the audio meeting resembled the latency graph of the video meeting at Figure 5-9. This increased both the latency and the jitter for audio packages significantly. This was unacceptable, since the audio communication is the fundamental part of a videoconferencing session. Although having the video feed of the remote party improves the quality of the communication, it is much more important to have a smooth and uninterrupted voice communication [ISAACS]. In addition, human ears are much more sensitive to the distortions on audio than human eyes are on the distortion of video. The distortions on audio cause much more discomfort. Therefore, we modified the routing algorithm at the broker and introduced another queue for audio packages. We have given priority to audio package routing over all other messages at the broker. When an audio package arrives at the broker, the broker routes this audio package first as long as it is done routing the current package. Therefore, it minimizes the routing

times of audio packages. While this approach ensured the quality of audio delivery, it did not reduce the quality of video communications significantly, as the following tests demonstrate.

5.4.1.4 The Impact of a Video Meeting on an Audio Meeting

First, we examine the impact of a video meeting on the performance and the scalability of an audio meeting. In this case, although both meetings were held concurrently, we gather the results from the audio meeting only. Table 5-7 shows the summary of these results. The columns are the same as of Table 5-5, but in this case the incoming traffic is 344 kbps, instead of 64 kbps. Similarly, the outgoing traffic is much higher than the single audio meeting case.

Figure 5-10 shows the comparison of average latencies of audio and video combined meetings with the average latencies of the single audio meetings. The average latency values for audio and video combined meetings are consistently larger than the average latency values of single audio meetings. In addition, the difference between these two graphs increases when the number of participants in meetings increases. However, the differences between these two graphs are not significant to affect the quality of the audio communication. When there are 600 participants, there is only 5ms difference. Therefore, the impact of the video meeting is not significant on the performance of the audio meeting.

As we can see, the latency values of the first clients are not the same as that of Table 5-5. It increases up to 5.8ms for 600 participants, since an audio package needs to wait at the broker for the completion of the routing of the currently routed package. As long as that package is routed, the audio package is routed to its destinations. This

results in a slight increase on the latency and the jitter values. This increase on the latency is limited to the maximum routing time of a package by the broker.

Table 5-7 Audio test results for single audio and video combined meetings

Number of clients	First latency (ms)	Middle latency (ms)	Last latency (ms)	Avrg. latency (ms)	Avrg. Jitter (ms)	Avrg. Late arrivals	In BW (kbps)	Out BW (kbps)
12	0.7	0.9	0.9	0.8	0.3	0	344	4.128
100	0.9	1.7	2.6	1.8	0.73	0	344	34.4
200	1.5	3.3	5.1	3.3	1.8	0	344	68.8
300	2	4.8	7.5	4.8	2.67	0	344	103.2
400	3.3	6.9	10.5	6.9	4.35	0	344	137.6
500	4.4	8.8	13.4	8.9	3.84	0	344	172
600	5.8	11.2	16.7	11.2	5.46	0	344	206.4

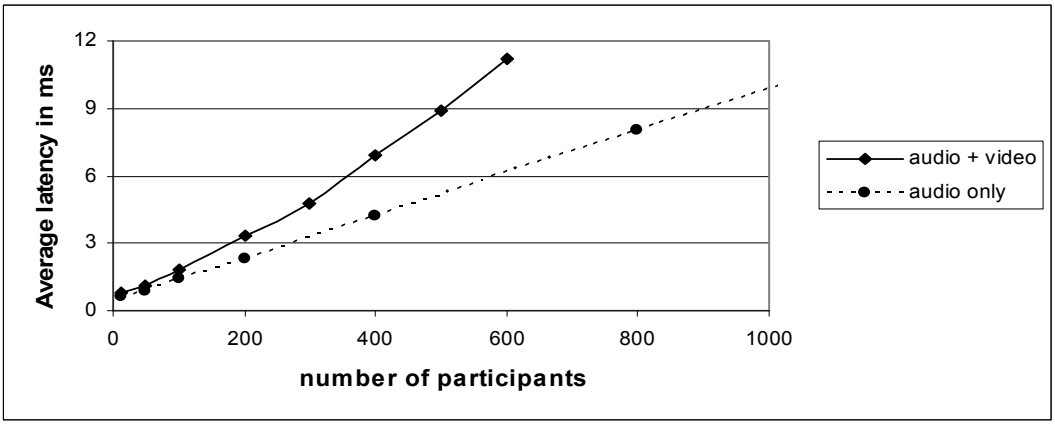

Figure 5-10 Comparison of avr. latencies for single audio meetings and audio+video meetings

Although the performance of the audio meeting is still very good for 600 participants, the performance of the video meeting gets unacceptable. Therefore, we have not gathered the results for larger size meetings.

5.4.1.5 The Impact of an Audio Meeting on a Video Meeting

Similar to the previous case where we examined the impact of a video meeting on an audio meeting, this time we examine the impact of an audio meeting on a video meeting. The setting is the same as the previous case but we gather the results from the video meeting only. Table 5-8 shows the results. All columns are the same as the columns of Table 5-6 of the single video meeting tests. The results of this test are also similar to the results of that test.

Figure 5-11 shows the average latency values of audio and video combined meetings, and the average latency values of the single video meetings for comparison. Although the latency values of audio and video combined meetings are higher than the average latency values of single video meetings, the difference is very small until the broker is overloaded in audio and video combined meeting tests. When there are 400 participants, the difference is only 5ms. In addition, there are no late arriving packages for 300 participants in audio and video combined meetings and there is %1.3 late arriving packages for the last user for 400 participants. This value is slightly higher than the single video meeting test in which the late arrival rate was %0.7 for 400 participants. Therefore, up to 400 participants can be supported in audio and video combined meetings.

On the other hand, the broker is overloaded much earlier in audio and video combined meetings than it is in the single video meetings, because of the extra load put

by the delivery of audio streams. Now 600 participants overload the broker compared to 1000 participants in single video meeting case.

Table 5-8 Video test results for single audio and video combined meetings

number of clients	first latency (ms)	middle latency (ms)	last latency (ms)	Avrg. Latency (ms)	Avrg. Jitter (ms)	Last Late arrivals	Avrg. Late arrivals	In BW Mbps	Out BW Mbps
12	1.3	1.4	1.5	1.4	0.5	0	0	344	4.13
100	3.7	4.6	5.7	4.7	2.1	0	0	344	34.4
200	8	10.1	12.2	10.1	5.4	0	0	344	68.8
300	11.3	14.2	17.2	14.2	7.5	0	0	344	103.2
400	18.1	22.1	26.1	22.1	10.9	% 1.37	% 1.25	344	137.6
500	26.7	31.7	36.8	31.8	13.5	% 5.5	% 4.7	344	172.0
600	169.2	175.3	181.4	175.3	16.2	% 59.9	% 58.2	344	206.4

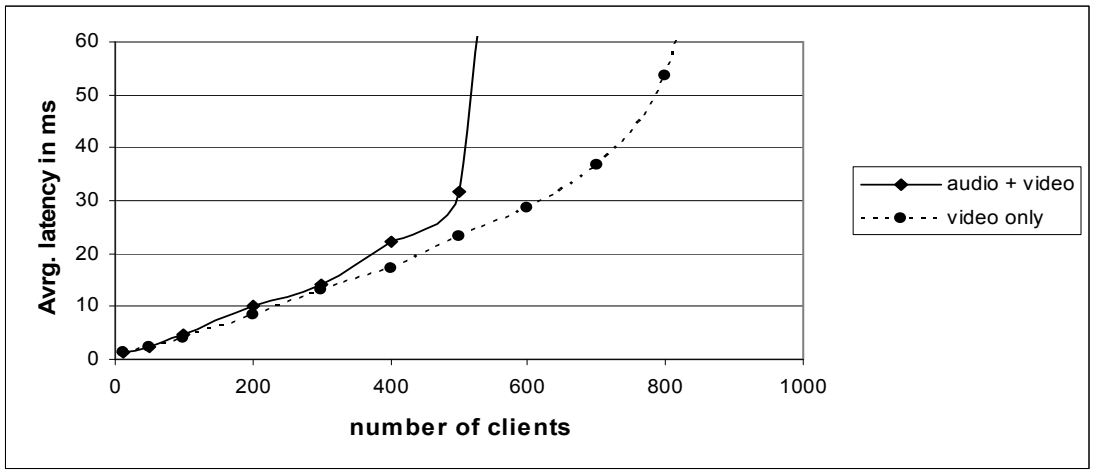

Figure 5-11 Comparison of avr. latencies for single video meetings and audio+video meetings

This test shows that the impact of an audio meeting on the performance of a video meeting is not significant, particularly when the number of participants in these meetings is not very high and the broker is not close to being overloaded. In audio and video combined meetings, the broker supports almost the same number of participants as in the case of single video meetings. The main reason for this is the better utilization of broker resources when there are two concurrent meetings.

We can summarize the performance tests for one broker with single meetings in this test environment. The broker supports up to 1500 participants in a single audio meeting. Its resources are utilized fully in the single audio meeting case when it is overloaded, because the audio packages are distributed evenly on time. The broker supports 400 participants in a single video meeting. Its resources are not utilized fully in the case of single video meetings when some packages started to arrive late, because of the full picture update frames. When an audio meeting is added to the single video meeting in the case of audio and video combined meetings, the broker resources are utilized much better and the broker can still supports up to 400 participants in audio and video combined meetings. Therefore, these tests show that a NaradaBrokering broker can be used to conduct single large size meetings for real-time videoconferencing with very good quality.

5.4.2 Multiple Meeting Tests

Similar to single meeting tests in the previous section, we tested the performance and the scalability of a single broker for three types of multiple concurrent meetings: multiple audio meetings, multiple video meetings, and multiple audio and

video combined meetings. We started from a few small size meetings and went up to many small size meetings until the broker can not deliver an acceptable performance.

We decided to have equal number of participants in all meetings to simplify both the testing and the analysis. We had 20 participants in all meetings. We gathered the results from 10 meetings when there are more than 10 meetings. These meetings had been chosen in no particular order. Since the transmitters of each meeting are independent of others, there is no best or worst performance meeting. We averaged these results to get the broker performance. All measuring transmitters and measuring clients were running on the same machine. Passive receivers were running on other machines in the Linux cluster.

5.4.2.1 Multiple Audio Meeting Tests

Multiple audio meeting tests are summarized in section 5.1.4 and the results are provided in Figure 5-5. As we have pointed out previously, when there are multiple meetings, the broker is utilized better than the single meeting case. Since, multiple streams arrive randomly distributed on time, the packages wait much less time on the broker on the average than single large scale meetings. Therefore, it provides better quality of service with smaller latency and jitter values.

5.4.2.2 Multiple Video Meeting Tests

The results of the multiple video meeting tests are summarized on Table 5-9 and Figure 5-12. Figure 5-12 shows the results of both multiple video meeting tests and single video meeting tests for comparison. The first column of Table 5-9 shows the number of total receivers on the broker. The second column shows the number of

meetings on each test. We started with 5 meetings and went up to 45 meetings. Average latency column shows the average latency of 10 receivers from 10 different meetings. The average jitter is also the average jitters of these 10 receivers. The average late arrivals are the average of the percentages of late arriving packages from 10 receivers. Incoming bandwidth is the amount of total data coming to the broker. Outgoing bandwidth is the amount of data the broker is sending out to receivers.

When we compare the average latency values of this test with the average latency values of the single video meeting test for the same number of total participants as it shown on Figure 5-12, we see that the latency values of the multiple video meeting tests are much smaller than the latency values of the single video meeting tests until the broker is overloaded. Similarly, the average jitter values of this test are much smaller than the average jitter values of the single video meeting tests. This conclusion is the same as the conclusion of the comparison of single audio meeting tests with the multiple audio meeting tests at section 5.1.4. That is, multiple meetings provide better utilization of the broker resources and result in smaller latency and jitter values.

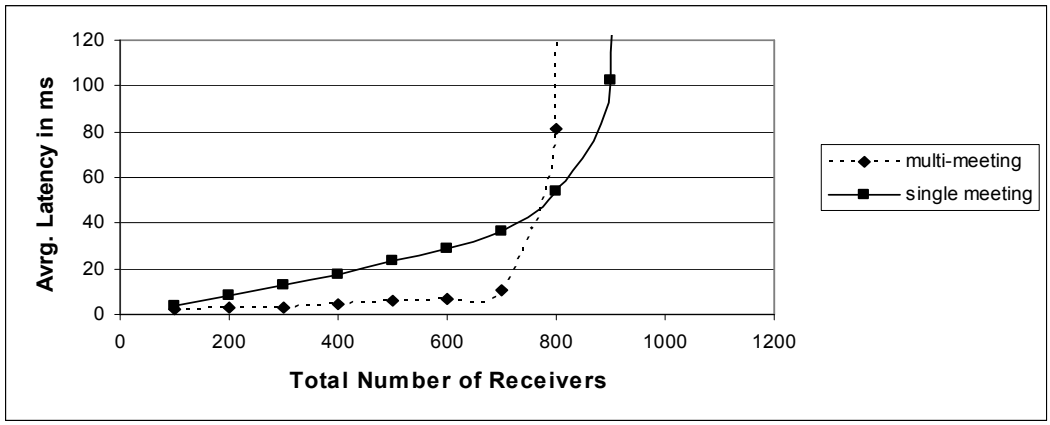

Figure 5-12 Single and Multiple Video Meeting Test Comparisons

Table 5-9 Multiple video meetings tests, each meeting having 20 users

Number of clients	Number of meetings	Avrg. latency (ms)	Avrg. Jitter (ms)	Avrg. Late arrivals	In BW (Mbps)	Out BW (Mbps)
100	5	2.25	0.68	0	1.43	28.7
200	10	2.74	0.85	0	2.87	57.4
300	15	3.17	0.86	0	4.30	86.1
400	20	4.54	1.1	0	5.74	114.8
500	25	5.94	1.3	0	7.17	143.5
600	30	6.8	1.37	0	8.61	172.2
700	35	10.64	1.52	% 0.7	10.04	200.9
800	40	81.1	1.8	% 19	11.48	229.6
900	45	2787	3.3	% 98	12.91	258.3

Similar to smaller latency and jitter values, multiple video meeting tests also provide much smaller late arrival ratios for video packages. There are no late arrivals for 600 receivers and there are very few late arrivals (%0.7) for 700 receivers. Therefore the broker can support 35 concurrent video meetings with 700 participants in total. This is much higher than the single video meeting case in which some packages started arriving late for 400 receivers. There are two reasons for the lower late arrival rates. These are the small meeting sizes and multiple concurrent meetings. We can illustrate the effects of small meeting sizes by calculating the latencies using equation (Eq. 16). It takes only 0.37ms to route a package to 20 participants in a meeting, and it takes 6.66ms to route 18 packages of a full picture update frame to the 20 participants of the same meeting. Therefore, the high number of packages in full picture update

frames does not result in late arrivals until the broker starts saturating. Figure 5-13 shows the latency graph of a participant from a meeting when there were 30 meetings with 600 participants in total. This graph shows that there are no peaks in latency values for full frames as it was the case in the single meeting case for 400 participants in Figure 5-9. In addition, since the packages of multiple streams arrive randomly on time on the broker, they utilize the broker resources more efficiently.

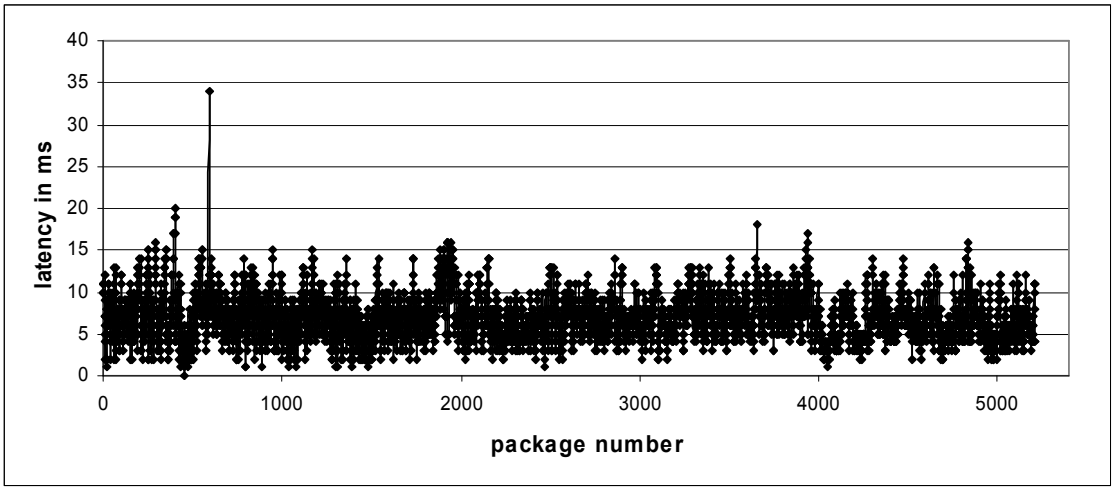

Figure 5-13 Latency graph for 30 video meetings with 600 participants in total

5.4.2.3 Multiple Combined Meeting Tests

In the case of multiple concurrent audio and video meetings, we have initiated the same number of audio and video meetings on the broker with 20 participants each. We have gathered two sets of results. One set of result is gathered from audio meetings while video meetings are active. Another set of results are gathered from video meetings while audio meetings are active. Table 5-10 shows the results gathered from audio meetings and Table 5-11 shows the results gathered from video meetings.

Table 5-10 Audio results from audio and video combined multi meeting tests

Number of Clients	number of meetings	Avrg latency (ms)	Avrg. Jitter (ms)	Avrg. Late arrivals	In BW (Mbps)	Out BW (Mbps)
200	10	1.7	0.5	0	1.755	35.1
400	20	2.5	0.9	0	3.51	70.2
600	30	3.3	2	0	5.265	105.3
800	40	4.9	2.2	0	7.02	140.4
1000	50	46.8	2.8	%16	8.775	175.5
1200	60	9287	6.6	%100	10.53	210.6

Table 5-11 Video results from audio and video combined multi meeting tests

Number of Clients	number of meetings	Avrg. latency (ms)	Avrg. Jitter (ms)	Avrg. Late arrivals	In BW Mbps	Out BW Mbps
200	10	2	0.7	0	1.755	35.1
400	20	2.62	0.85	0	3.51	70.2
600	30	5.25	1.3	0	5.265	105.3
800	40	6.5	1.56	0	7.02	140.4
1000	50	76.2	1.96	%23	8.775	175.5
1200	60	9421	4.12	%100	10.53	210.6

The columns of these two tables are the same as the columns of Table 5-9 from the previous multiple video meeting tests. However, first columns of these tables show the total number of participants in audio + video meetings. Half of these participants are audio meeting participants and the other half are video participants. Similarly the

second columns show the total number of audio + video meetings. Half of them are audio meetings and the other half are video meetings.

As it can be seen from these results, until the broker is overloaded around 50 meetings or 1000 participants, the average latency is very small. In addition, there are no late arriving packages until 800 participants. These results demonstrate that 40 meetings (20 audio and 20 video meetings) can be conducted simultaneously on this broker with excellent quality. In addition, these results show that the routing of audio streams does not degrade the performance of video streams significantly. Although audio latencies are better than video latencies, both provide very low latency for videoconferencing.

As we pointed out previously, in single meeting tests the maximum number of supported video participants remained unchanged when the audio meeting is added to the single video meeting. One broker supported 400 participants in both single video meeting case, and audio and video combined meeting case. Therefore, adding an audio meeting did not reduce the number of supported participants in the single video meeting. However, in the case of multiple audio and video combined meetings, having audio meetings with video meetings reduced the maximum number of supported video meetings significantly. Previously, the broker was able to support 35 concurrent video meetings, now it can only support 20 video and 20 audio meetings. Therefore, the number of supported video meetings is reduced almost by half. The main reason for this difference is that, in single video meetings, the broker was not utilized efficiently. Therefore, adding the audio meeting increased the efficiency of the broker and still supported the same number of video participants. However, in multiple video meetings

case, the broker was already utilized efficiently and adding the audio meetings reduced the number of supported video meetings.

In summary, a NaradaBrokering broker provides excellent quality audio and video delivery services to high number of participants for both single and multiple meetings. Hundreds of participants can be easily supported by one broker. The number of supported users and the quality of media service delivery can be further increased by running multiple servers. Now, we will investigate the performance and the scalability the brokering network in distributed settings.

5.5 The Performance Tests for Distributed Brokers

In this section, we evaluate the performance and the scalability of NaradaBrokering broker network in distributed settings with multiple brokers. Similar to single broker tests, we evaluate the performance and the scalability of the brokering network for both single large size meetings and multiple smaller size meetings. These tests provide valuable information about how to utilize the resources of the broker network in the most efficient manner. However, we first evaluate the package routing algorithm in NaradaBrokering and propose some improvements.

5.5.1 Inter-Broker Delivery Priority for Distributed Brokers

In this section, we evaluate the package routing algorithm of NaradaBrokering network in distributed setting. This evaluation reveals a weakness in the algorithm that limits the scalability of the broker network. We propose a modification to this algorithm and perform some tests to show the results.

A NaradaBrokering broker routes received packages in the following manner. First, every received package is placed into a first-come-first-serve queue. The routing thread picks up the first arrived package from this queue, calculates the destinations and transmits it to all subscribers of that topic. The routing thread continues to route packages one by one as long as there are packages in the queue. In distributed settings, if a package needs to be delivered to other brokers as well, the routing thread first sends the package to other brokers and then to local subscribers. This is to avoid introducing more delays to the transit time of packages that need to travel multiple brokers.

When we started testing the distributed brokers for a single meeting, we have observed that an overloaded broker can introduce significant delays to packages that are routed through it to other brokers with the given algorithm above. Although other brokers might not be overloaded, their subscribers can still be severely affected by the load of an overloaded broker along the path from producer to the consumer. In addition, the travel times for packages increases significantly when they go through multiple brokers. Each broker along the path can introduce unnecessary delays. This limits the number of brokers a package can travel. Therefore, it limits the scalability of the broker network considerably. Here we demonstrate this by providing the following test case.

We have set up a two broker NB cluster (Figure 5-14). We initiated one video meeting on them. We loaded the first broker with 400 participants. The second broker had only 6 participants. One video stream is published to the meeting through the first broker. We gathered the results from 6 receivers of the first broker and 6 receivers of the second broker. Six receivers of the first broker were the first 2, middle 2 and the last 2 subscribers. Therefore, they provided best, middle and worst results.

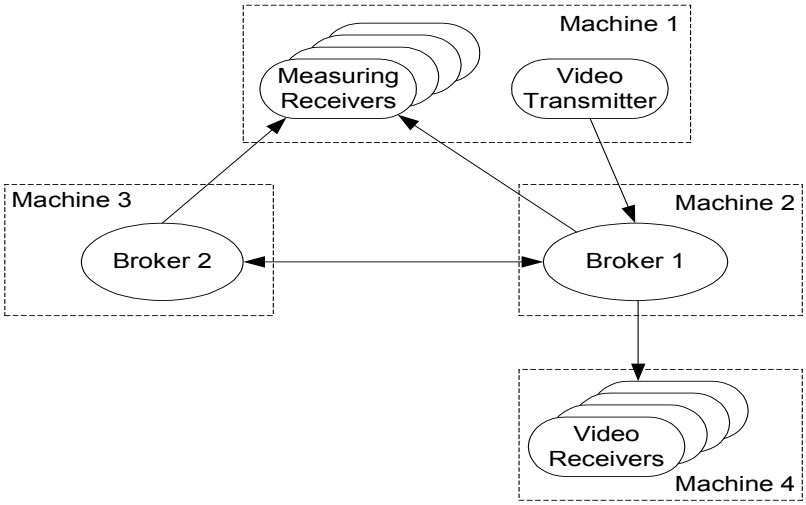

Figure 5-14 Single video meeting setup on two brokers

Table 5-12 shows the average latencies of the 5610 video packages transmitted. The latency results of the receivers of the second broker are very close to the latency results of the first receiver of the first broker. They are slightly higher, because the packages travel to the second broker through the first broker to reach to the receivers. Otherwise, the first broker delivers each package first to the second broker and then to its local subscribers.

This test shows that the first broker introduces significant delays to the transit times of video packages transmitted to the second broker. Although the second broker has very few subscribers, it is still bounded by the first broker. The subscribers of the second broker can not get a service better than the first receiver of the first broker. The reason for this is the first-come-first-serve queue in the first broker. When there are multiple packages in this queue, the later ones need to wait the earlier ones to be routed to all 400 local subscribers. This introduces significant delays to packages that will be

delivered to the second broker. This can be eliminated by a mechanism that will route packages first to other brokers in the system without waiting local subscribers to be served. This can be implemented by introducing another first-come-first-serve queue and another thread to the broker.

Table 5-12 Latency values for single video meeting test for two brokers

	First Receiver (ms)	Middle Receiver (ms)	Last Receiver (ms)	Avrg. (ms)
Broker 1	15.83	20.19	24.55	20.20
Broker 2	16.07	16.12	16.18	16.13

5.5.1.1 Double Queuing Algorithm

Double queuing algorithm separates inter-broker delivery of packages from local client deliveries. It aims to introduce minimum delays to packages that will be routed to other brokers in the system. In addition to an additional queue, it also introduces another routing thread. Figure 5-15 depicts this algorithm. Received packages are first placed into the first queue. The first thread picks up a package from this first-come-first-serve queue and delivers it to other brokers in the system if there is any. It hands it over to the second thread by placing it into the second queue after finishing the inter-broker routing. Then it continues to route the next package in the first queue without waiting the second thread to serve the local clients. The second thread continues to serve the local clients as long as there are packages in the second queue. Since these two threads have similar priorities they work concurrently.

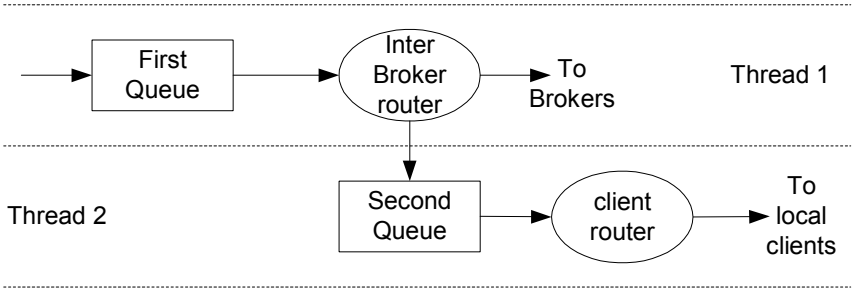

Figure 5-15 Double queuing algorithm

We should remember that there are very few brokers to route a package in a typical distributed NB network. A broker is connected to less than 5 other brokers in most cases. Therefore, inter-broker routing will take very small amount of time compared to the delivery of packages to high number of local clients.

We should also note that we implement two separate queues for both of these queue layers. As we have outlined previously in section 5.4.1.3, we give priority to audio package routing in brokers. Therefore, there is one audio queue and another for all other packages. The threads first route the audio packages if there is any, then they route the other packages.

We have repeated the single video meeting test for two brokers with the same setting for the double queuing algorithm. Table 5-13 shows the results. As it can be seen, the latency results of the receivers of the second broker are very small. The only overhead introduced by the first broker is the overhead of routing packages to another broker. The high number of clients in the first broker does not impact the performance

of the receivers of the second broker. This new algorithm eliminates cases where an overloaded broker severely affects the performance of the subscribers of other brokers.

Table 5-13 Latency values for single video meeting test for two brokers

	First Receiver (ms)	Middle Receiver (ms)	Last Receiver (ms)	Avrg. (ms)
First broker	16.07	20.53	24.92	20.52
Second broker	1.41	1.50	1.62	1.52

Another very important advantage of the double queuing algorithm is to enable the brokering network to grow to high number of brokers. As we can see from the latency results at Table 5-13, traveling of packages through a relatively loaded broker takes only around 1ms, compared to 15ms previously. Now, even going through 10 brokers for a package only introduces around 10ms of overhead when the transmission delays between the brokers are ignored. This makes it possible for packages to travel many brokers along the way from sources to destinations. Therefore, it enables the broker network to grow in size to large numbers.

In addition, by using the double queuing algorithm, the load on a large size meeting can be distributed among multiple brokers easily. Since the brokers add minimum delays to packages traveling to other brokers, it makes it very efficient to add extra brokers into the system to support more users or to provide better quality of service. To demonstrate this, we have conducted another test similar to previous two cases in which each broker had 200 participants. Table 5-14 shows the results of this test. As it can be seen, the latency values of the first broker and the second broker is

very similar. The values of the second broker are slightly higher because the packages travel through the first broker to reach to the second broker. When we compare the results of Table 5-14 and Table 5-13, we see that the distribution of 400 participants into two brokers reduces the average latencies by fifty percent. This result illustrates that more brokers can be introduced into the system to provide better quality of service. Similarly, more brokers can be added into the system to provide services to higher number of participants.

Table 5-14 Latency values for single video meeting test for two brokers

	First Receiver (ms)	Middle Receiver (ms)	Last Receiver (ms)	Avrg. (ms)
First broker	7.94	10.16	12.37	10.16
Second broker	8.24	10.43	12.67	10.45

One disadvantage of this double queuing algorithm might be the extra overhead put on the delivery times of the clients of the first broker. When we compare the latency values of the receivers of the first broker in Table 5-12 and Table 5-13, we see that the latency values of Table 5-13 are slightly higher than the latency values of Table 5-12 (0.3ms on the average). This slight increase can be negligible given the advantage of this solution. On the other hand, this disadvantage might be more apparent when there are more concurrent meetings, and more streams are delivered to other brokers before serving the local subscribers. To investigate this, we have setup a testing environment with multiple brokers and many concurrent small size meetings.

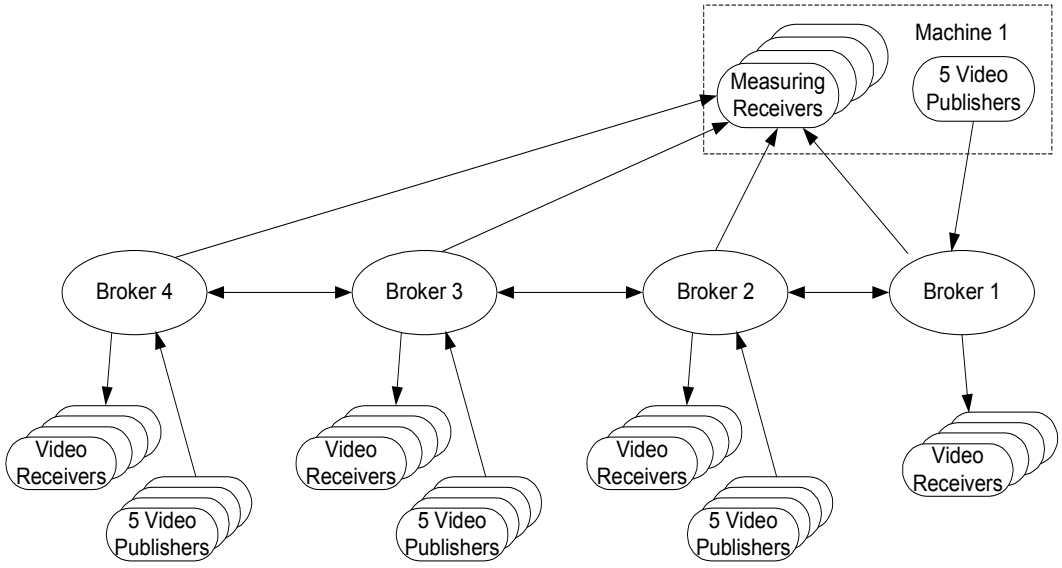

Figure 5-16 Multi broker test setting for multiple meetings

The testing scenario (Figure 5-16) included 4 brokers connected as a chain. We have setup 20 concurrent meetings. Five participants joined each meeting through every broker. Therefore the size of each meeting was 20. One video stream is published to every meeting. Five of the video publishers were attached to every broker. We have gathered the results from three meetings. All the transmitters of these three meetings were publishing their streams through the first broker in the chain. For each of these three meetings, we calculated the latencies of every video package and averaged them. We have repeated this test for both single queuing and double queuing algorithms. Table 5-15 shows the results of this test with single queuing algorithm and Table 5-16 shows the results of this test with double queuing algorithm. As it can be seen from these two tables, double queuing algorithm provides slightly higher latency values. On the average, it delivers packages 0.26ms later than the single queuing algorithm.

However, this algorithm does not put significant overhead even in multiple small size meetings. We should also note that the average latency for the fourth broker is slightly lower for the double queuing case. This is expected since the last broker does not route the video packages to any other broker for measuring streams.

Table 5-15 Multiple video meeting tests for distributed brokers with single queuing

	Meeting1	Meeting2	Meeting3	Avrgs.
broker1	3.2	2.65	3.56	3.13
broker2	5.34	4.1	5.81	5.08
broker3	7.43	5.62	7.91	6.98
broker4	7.93	5.87	8.28	7.36
average				5.64

Table 5-16 multiple video meeting tests for distributed brokers with double queuing

	Meeting1	Meeting2	Meeting3	Avrgs.
broker1	4.07	3.98	3.53	3.86
broker2	6.05	5.83	4.61	5.49
broker3	7.78	7.35	5.79	6.97
broker4	8.31	7.51	6.01	7.27
average				5.90

In summary, with double queuing algorithm, a NaradaBrokering broker puts very small amount of overhead to packages that are traveling to other brokers. This makes it possible for packages to travel many hops along the way from sources to destinations. Therefore, the broker network can grow to large sizes by providing

excellent scalability. In addition, this algorithm prevents an overloaded broker to affect the performance of clients of other brokers that are connected to the overloaded broker.

5.5.2 Single Meeting Tests for Distributed Brokers

We first test the performance and the scalability of the broker network for single video meetings. We analyze the results for the quality of services provided and compare them with the single broker tests. At the end, we also provide the test results for the single audio meetings.

In addition to the Linux cluster that we used for single broker tests, we had access to another Linux cluster for distributed broker tests. Both clusters had 8 nodes. While the nodes of the previous cluster had 2.4GHz Dual Intel Xeon CPU and 2 GB of memory, the nodes of the second cluster had slightly better computing power. They had 2.8 GHz Dual Intel Xeon CPU and 2GB of memory. Both clusters had gigabit network connection among their nodes. The network bandwidth between the two clusters was 100Mbps.

We set up a distributed broker network with 4 brokers (Figure 5-17). We connected the brokers as a chain to get the maximum travel distances for packages in order to observe the impact of package travels through many hops. We ran two brokers in each Linux cluster. In other machines we ran the transmitter and receiver clients. All brokers belonged to the same NB cluster. Broker1 was the cluster controller and it was the first broker to be started. Broker2 was started as the second broker and connected to Broker1. Similarly, Broker3 was attached to Broker2 and Broker4 was attached to Broker3. The transmitter client was publishing the video stream through Broker1.

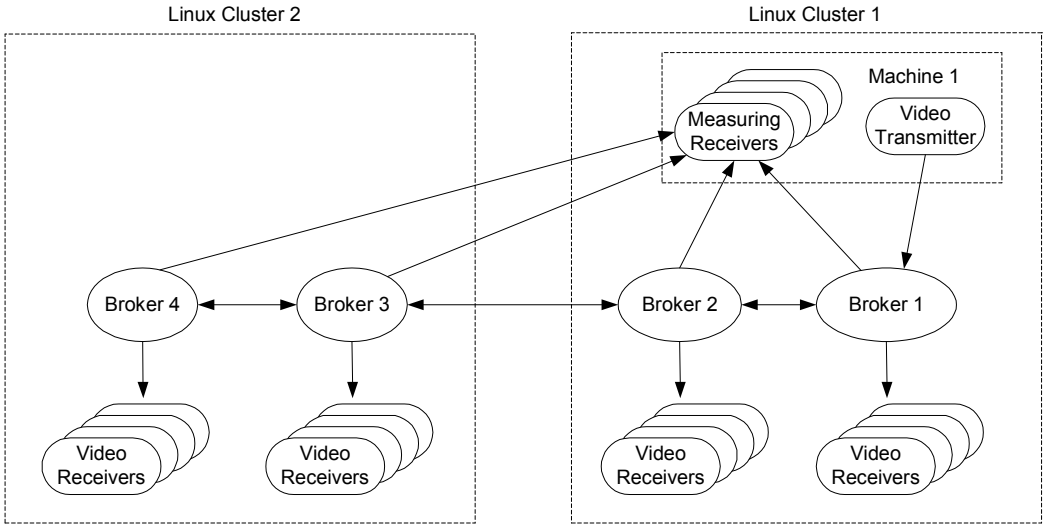

Figure 5-17 Single meeting tests with multiple NB brokers

Similar to single video meeting tests with one broker, a user published the video stream on a topic and all other participants received it by subscribing to that topic. The receivers were evenly distributed among the brokers and equal number of receivers joined the meeting through each broker. We gathered the results from the first and the last users of every broker. Table 5-17 shows the results of these tests. Every row shows a test case. Each test case has different number of receivers in the meeting. First column shows the number of receivers on each broker. We start from 3 receivers and go up to 900 receivers in each broker. The second column shows the total number of receivers at four brokers in a meeting. Remaining two blocks of results show the average latency values of the first and the last receivers in each broker for the 5610 video packages exchanged.

Table 5-17 Single video meeting latency results for distributed brokers

users per broker	Users in total	Latencies of first receivers (ms)				Latencies of last receivers (ms)			
		B1	B2	B3	B4	B1	B2	B3	B4
3	12	1.34	1.7	2.86	3.21	1.41	1.69	2.97	3.46
100	400	4.34	4.8	5.45	5.86	6.58	7.04	7.34	7.73
200	800	8.06	8.44	8.58	9.03	12.57	12.93	12.39	12.79
300	1200	12.04	12.5	11.85	12.37	18.78	19.25	17.55	18.03
400	1600	16.45	16.94	15.26	15.95	25.47	26.02	22.84	23.55
500	2000	21.43	22.38	19.07	19.92	32.61	33.72	28.53	29.38
600	2400	28.5	29.77	23.88	24.1	41.88	43.31	35.36	35.45
700	2800	45.27	50.54	30.75	30.4	61	66.58	44.21	43.66
800	3200	127.6	119.9	42.51	41.73	146	138.17	57.87	56.99
900	3600	665.3	798.4	86.9	72.99	685.7	818.98	104.5	90.17

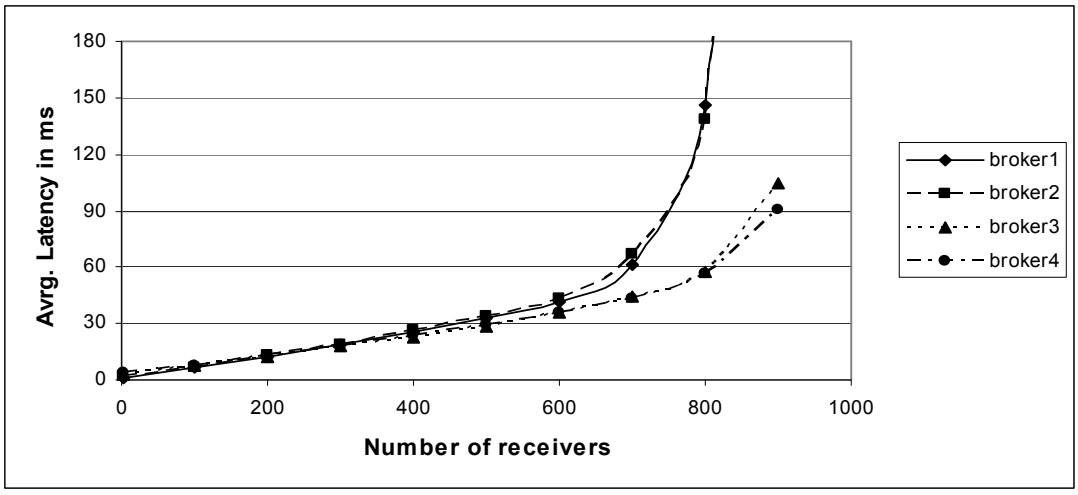

Figure 5-18 Latencies of the last receivers from 4 brokers in single video meetings

The latency results of broker1 and broker2 are very similar to the latency results of the single video meeting tests for one broker. They are slightly worse because the video stream is first passed to the next broker. Otherwise, the values and the characteristics of the latency function are the same.

As it can be seen from Table 5-17 and from the latency graph of the last receivers (Figure 5-18), Broker3 and Broker4 perform better than the first two brokers. Although the difference is not apparent in small size meetings, it becomes more visible when the size of the meetings increases. The main reason for this is the superior CPU power of the machines in the second Linux cluster. Since those machines have more computing power, they can serve more packages in the same amount of time and introduces smaller delays to the transit times of packages.

Table 5-18 Percentages of late arriving packages for last users in single video meetings

Users per broker	Users in total	Broker1 late arrivals (%)	Broker2 late arrivals (%)	Broker3 late arrivals (%)	Broker4 late arrivals (%)
3	12	0	0	0	0
100	400	0	0	0	0
200	800	0	0	0	0
300	1200	0.29	0.27	0	0.02
400	1600	1.87	1.92	0.73	0.73
500	2000	4.01	4.43	2.41	2.4
600	2400	7.15	7.55	4.66	4.39
700	2800	16.64	19.78	7.59	7.1
800	3200	53.59	51.56	14.28	13.67
900	3600	93.54	93.85	37.84	31.31

This test demonstrates that the NB network scales very well for a single video meeting. The capacity of the broker network can be increased almost linearly by adding new brokers into the system. Table 5-18 shows the percentages of late arriving packages for the last users in each broker. The percentages of late arriving packages are %1.9 for the first two brokers when there are 400 participants. Therefore, these two brokers support less than 400 participants. The percentages of late arriving packages are %0.7 for the last two brokers. Therefore, they can support more than 400 participants. Four brokers support around 1600 participants in total. The overhead of traveling packages through 4 brokers is very small and adding new brokers into the system increases the number of supported users almost linearly.

Table 5-19 Average jitter values for single video meeting tests with four brokers

Users per broker	Users in total	Broker1 Jitter (ms)	Broker2 Jitter (ms)	Broker3 Jitter (ms)	Broker4 Jitter (ms)
3	12	1.1	0.93	0.92	1.03
100	400	2.85	2.92	2.45	2.89
200	800	5.92	5.91	5.01	5.61
300	1200	8.97	9.02	7.62	8.24
400	1600	12.07	12.21	10.13	10.97
500	2000	14.96	15.22	12.61	13.6
600	2400	17.98	18.22	15.37	15.78
700	2800	21.21	21.68	18.08	18.49
800	3200	24.83	24.66	20.65	20.87
900	3600	27.68	27.78	23.77	23.52

The average jitter values of receivers from each broker are given at Table 5-19. The jitter values of receivers from the last two brokers are slightly better than the first two brokers, because of the superior hardware of the second Linux cluster. These results show that the travel of packages through multiple brokers does not result in an increase in the jitter. Because the overhead introduced by the travel from one broker to another is very small and almost constant for consecutive packages in the stream.

Similar to the single video meetings test, we also conducted the same experiment for a single audio meeting. The results are shown at Table 5-20. The results of this test confirm our conclusions from the video tests.

Table 5-20 Single audio meeting test results for distributed brokers

users per broker	users in total	latencies of first receivers (ms)				latencies of last receivers (ms)			
		B1	b2	b3	B4	B1	b2	b3	b4
3	12	0.69	0.98	1.79	2.02	0.7	1	1.83	2.04
100	400	0.62	0.9	1.74	1.92	2.56	2.83	3.34	3.56
200	800	0.65	0.96	1.78	1.98	4.65	4.98	5.17	5.35
400	1600	0.66	0.96	1.87	2.07	8.76	9.29	8.65	8.9
800	3200	0.68	1.04	2.11	2.31	16.98	17.09	15.75	16.25
1200	4800	0.68	0.99	1.86	2.08	25.17	25.64	22.39	22.84

In summary, these tests demonstrate that NaradaBrokering broker network scales well in distributed settings when delivering audio and video streams to high number of participants in a meeting. The scalability of the system increases almost linearly by the number of brokers. The overhead of going through multiple brokers for a stream is not significant, since inter-broker routing has priority over local client

routings. This makes NaradaBrokering a scalable solution for the delivery of audio/video streams to high number of participants.

5.5.3 Multiple Meeting Tests for Distributed Brokers

The behavior of the broker network is more complex when there are multiple concurrent meetings compared to having a single meeting. Having multiple meetings provide both opportunities and challenges. As we have seen in the single broker tests with multiple video meetings in section 5.4.2.2, conducting multiple concurrent meetings on a broker can increase both the quality of the service provided to clients and the number of supported users. This can also be achieved in multi broker settings as long as the size of these meetings and the distribution of clients among brokers are managed appropriately. If the sizes of meetings are very small and the clients in meetings are scattered around the brokers, then the broker network can be utilized poorly. The best broker utilization is achieved when there are multiple streams coming to a broker and each incoming stream is delivered to many receivers. If all brokers are utilized fully in this fashion, multi broker network provides better services to higher number of participants. To investigate this, we conducted multiple video meeting tests with two different meeting sizes.

We used the same broker organization scheme as the single meeting tests in the previous section. There were four brokers connected as a chain. In this case, all brokers were running in the same Linux cluster that has 8 identical nodes with 2.8 GHz Dual Intel Xeon CPU and 2GB of memory. We also used five other Linux machines that are connected to this cluster with a gigabit bandwidth to run the passive receiver clients.

First, we tested with multiple meetings each having 20 receivers. One client was publishing the video stream on a broker and 20 clients were receiving it in every meeting. We distributed the clients of each meeting among brokers evenly. 5 clients joined each meeting through each broker. We also distributed the video transmitters evenly among the brokers. There were equal numbers of transmitters publishing video streams to each broker. For example, when there were 12 meetings, 3 transmitters were publishing video streams to each of the brokers. All these transmitters were independent of others.

We collected the performance data from three meetings. We started other meetings in advance and they were going on when we started the ones that we test. The publishers of these three meetings were publishing their streams through the first broker. For each of these three meetings, we collected the results from 4 receivers, each one getting the stream from a different broker.

Table 5-21 shows the average latency values of three meetings. Each row of the table shows a test case with different number of meetings. First column shows the number of meetings on the broker network for that test. Second column shows the total number of receivers in all meetings. Third column shows the average latencies of three measuring receivers that receive the streams from the first broker. Broker2 column shows the average latency results from the second broker in the chain. The next columns show the latency values from third and fourth brokers, respectively.

The other three tables below show the other parameters that we tested. Table 5-22 shows the percentages of lost packages, Table 5-23 shows the percentages of late arriving packages, and Table 5-24 shows the jitter values.

Table 5-21 Average latencies for multiple video meetings each having 20 participants

number of meetings	All users	Broker1 (ms)	Broker2 (ms)	Broker3 (ms)	Broker3 (ms)
12	240	1.85	2.99	3.84	4.11
24	480	3.30	4.24	5.32	5.87
48	960	2.98	5.04	6.90	8.20
72	1440	4.83	13.66	17.03	17.52
96	1920	5.76	25.55	52.77	47.34

Table 5-22 Average loss rates for multiple video meetings each having 20 participants

number of meetings	All users	Broker1 (%)	Broker2 (%)	Broker3 (%)	Broker3 (%)
12	240	0.00	0.00	0.00	0.00
24	480	0.01	0.00	0.00	0.00
48	960	0.00	0.06	0.11	0.22
72	1440	0.02	1.20	1.60	1.63
96	1920	0.13	6.41	19.62	19.68

Table 5-23 Average late arrivals for multiple video meetings each having 20 participants

number of meetings	All users	Broker1 (%)	Broker2 (%)	Broker3 (%)	Broker3 (%)
12	240	0.00	0.03	0.01	0.01
24	480	0.00	0.00	0.03	0.03
48	960	0.00	0.11	0.09	0.17
72	1440	0.00	0.01	0.03	0.04
96	1920	0.00	0.13	2.79	0.91

Table 5-24 Average jitters for multiple video meetings each having 20 participants

number of meetings	All users	Broker1 (ms)	Broker2 (ms)	Broker3 (ms)	Broker3 (ms)
12	240	0.81	1.10	1.37	1.46
24	480	1.39	1.44	1.65	1.72
48	960	1.84	2.53	2.82	3.00
72	1440	1.70	3.04	3.52	3.51
96	1920	1.69	3.70	5.28	5.52

Since the publishers are publishing the streams through the first broker, the latency values for the first broker are the smallest. Latency values increase when the streams travel more hops along the way from the first broker to the last. The broker network provides excellent quality communication when there are less than 72 meetings. The latency values and jitters for all brokers are very small. There are minor package losses and late arriving packages. For 72 meetings, the latency values and jitters are still very small. There is also very few late arriving packages. However, there are more lost packages. When there are 96 meetings, significant amount of packages are lost. Therefore, the broker network can support up to 72 meetings or 1440 participants in total. This number is slightly smaller than the single video meeting case, in which the broker network was able to support 1600 participants.

When we compare the scalability of the broker network with the scalability of the single broker in multiple meeting tests in section 5.4.2.2, the number of supported participants increased two times. The single broker supported 700 participants in 35 video meetings, each having 20 users. In this test, four brokers supported 1440

participants in 72 meetings, each having 20 users. As we can see, the increase on the number of brokers did not result in a linear increase on the number of supported participants. There are two reasons for this. First one is the overhead of inter-broker stream delivery in distributed setting. Now, the brokers deliver streams not only to clients but also to other brokers. The second one is the smaller number of participants in each broker for each meeting. Each incoming package is delivered to only 5 users in the distributed setting, while it was delivered to 20 users in the single broker case. This test shows that the distribution of users in small size meetings among multiple brokers reduces the scalability of the system. When meeting sizes are small, it would be better to avoid distribution of users among brokers if possible. On the other hand, the distribution of users in large scale meetings among multiple brokers increases the scalability and the quality of the service as we pointed out in double queuing algorithm section (section 5.5.1).

Since the small number of participants joining meetings through each broker is reducing the scalability and the quality of the provided service, we tested with a larger meeting size to observe the difference. This time, 10 participants joined each meeting from each broker. Therefore, the sizes of meetings were 40. All other aspects of the test were the same as the previous test.

Similar to previous test results, below tables show the measured parameters. Table 5-25 shows the average latency values of three measured meetings. Table 5-26 shows the percentages of lost packages, Table 5-27 shows the percentages of late arriving packages, and Table 5-28 shows the jitter values.

Table 5-25 Average latency values for multiple video meetings each having 40 participants

number of meetings	All users	Broker1 (ms)	Broker2 (ms)	Broker3 (ms)	Broker3 (ms)
40	1600	3.34	6.93	8.43	8.37
48	1920	3.93	8.46	14.62	10.59
60	2400	9.04	170.04	89.97	25.83

Table 5-26 Average loss rates for multiple video meetings each having 40 participants

number of meetings	All users	Broker1 (%)	Broker2 (%)	Broker3 (%)	Broker3 (%)
40	1600	0.00	0.00	0.00	0.00
48	1920	0.12	0.29	0.50	0.50
60	2400	0.16	1.30	2.51	2.82

Table 5-27 Average late arrivals for multiple video meetings each having 40 participants

number of meetings	All users	Broker1 (%)	Broker2 (%)	Broker3 (%)	Broker3 (%)
40	1600	0.00	0.00	0.00	0.00
48	1920	0.03	0.14	0.57	0.11
60	2400	0.00	53.02	36.48	0.69

Table 5-28 Average jitters for multiple video meetings each having 40 participants

number of meetings	All users	Broker1 (ms)	Broker2 (ms)	Broker3 (ms)	Broker3 (ms)
40	1600	1.15	2.20	1.99	2.10
48	1920	1.47	2.12	2.57	2.27
60	2400	2.42	4.62	4.81	4.60

In this case, the number of supported clients increased significantly. Now, 48 meetings with 1920 participants in total are supported with excellent quality, compared to 1440 participants in the previous test with meeting sizes of 20. In addition, the quality of the service provided by the broker network also increased considerably. The average latency and jitter values are much lower. The late arriving packages and losses are very small, too. The main reason for the better performance is the better utilization of the broker network. Now, there is less stream exchange among brokers and each incoming stream is delivered to more participants by every broker.

The scalability and the quality of service provided in this case are also much better than the single video meeting case on multiple brokers. Compared to the 1600 total users, now 1920 participants are supported. In addition, the latency and jitter values are much smaller. For 1600 participants, the latency values are less than 10ms, it was around 23ms for the single video meeting case for the last receiver. There is also a big difference in jitter values. While it is around 2ms for this case, it was more than 10ms for the single video meeting case. Therefore, this test demonstrates that similar to the single broker tests, it is possible to better utilize the distributed broker network by

having multiple video meetings than by having a single video meeting as long as the distribution of clients among brokers and the meeting sizes are chosen properly.

We should also note the big difference among latency values from four brokers when there are 60 meetings (Table 5-25). While the latency values from the first and the last brokers are relatively small, the latency values from the second and the third brokers are much higher. The main reason for this difference is the load on brokers. Since the broker network is organized as a chain, the middle two brokers in the chain delivers more streams among brokers. While the brokers on the edge delivers only the streams that are directly published on them by the clients to the next broker, the middle brokers both act as a bridge between the brokers on both sides and transmit the streams that are directly published on them to two other brokers on both sides. Therefore, this broker organization scheme puts more loads on the middle brokers. Other broker organization schemes can be explored such as ring or full mesh to avoid such disproportionate load.

In summary, there are two main benefits of having multiple brokers. The first one is the support for higher number of users. In the single meeting case, the number of supported users increases almost linearly by adding new brokers into the system. In multi meeting case, a similar increase can be achieved as long as the distribution of clients among brokers is done properly. The second benefit of having multiple brokers is to provide better quality service with smaller latencies, jitters and loss rates to clients. Particularly the quality of stream delivery for large size meetings can be improved significantly by distributing clients among multiple brokers.

5.6 Wide-Area Media Delivery Tests

In this section, we investigate the delivery of audio/video streams to geographically distant clients. We measure the quality of services provided in real life videoconferencing settings where clients can be scattered around the world. We will examine the performance results with single and multiple brokers. The results will also provide guidelines to distribute brokers in geographically distant settings.

We had access to machines at five different sites. In addition to the two Linux clusters that we used for the previous distributed broker tests, we had access to machines at three more locations. The previous two Linux clusters were in the same town (Bloomington, IN) at separate departments. The first cluster was in Community Grids Labs at IU (CGL) and the second cluster was in the department of Computer Science (CS) at IU. For these tests, both of these sites were connected to Internet2 with a gigabit bandwidth. The other three sites were in geographically distant locations. One site was in Syracuse University at Syracuse, NY. The other site was in Florida State University (FSU) at Tallahassee, FL. The last site was in Cardiff University in Cardiff, UK. Therefore, four sites were in USA and one site was in United Kingdom. All five sites were in universities. These sites had various network capacities. Syracuse had 90 Mbps download and 6.5 Mbps upload speed to Internet2. FSU had 90 Mbps download and 5 Mbps upload speed to Internet2. Cardiff had the maximum speed of 10 Mbps for both download and upload speeds.

We tested the delivery of video streams because of its nonlinear bandwidth over time and its requirement for higher bandwidth. We have used the same video stream that we used for other tests.

5.6.1 Video Meeting Tests with one Broker

First, we tested the quality of service provided to remote participants in a video meeting with one broker. We tested with varying number of clients at remote sites to assess the quality of the services for different number of participants. The broker was running at the CS site in Indiana and equal numbers of participants were running at the four other sites (Figure 5-19). A client running in the same site as the broker published the video stream on the broker. We measured the latencies, jitters and loss rates for the first and the last clients at each site. We calculated the latency values for each video package. We recorded the sent times of packages on the transmitter machine and received times of the packages on the receiver machines. We also measured the clock differences between the transmitter and the receiver machines to calculate the latencies. There can be a few millisecond discrepancies on latency values because of the difficulties on determining the exact clock differences among remote machines.

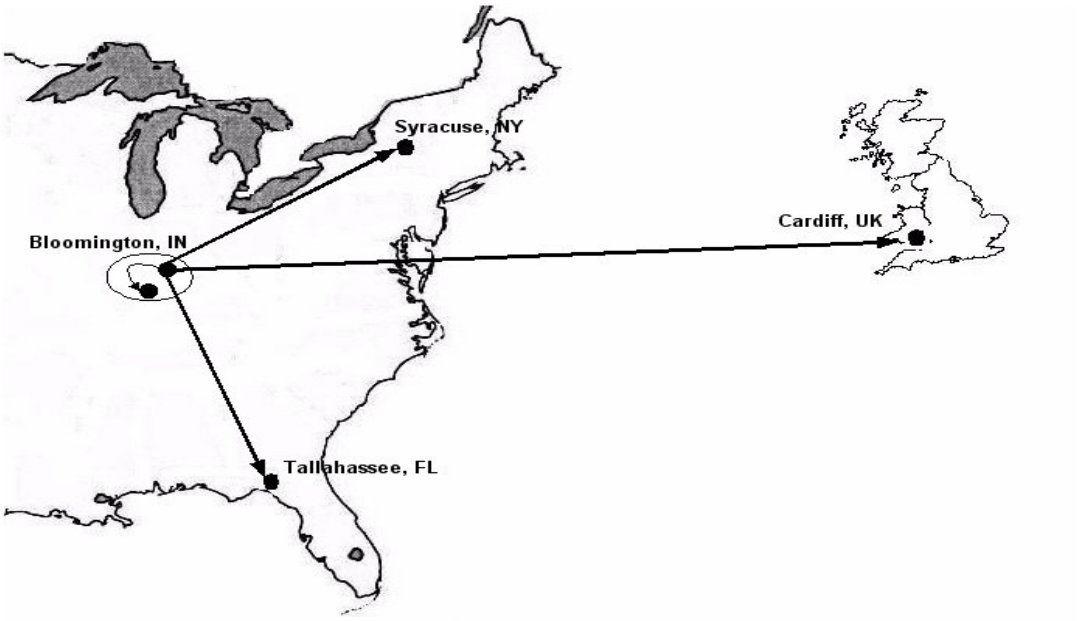

Figure 5-19 Delivery of video streams to remote clients

Table 29 shows the latency values for the first and the last users in every site for each test case. First column shows the number of participants on each site. The second column shows the total number of participants in the meetings. Third column shows the latencies of the first client at CGL in Indiana. The next three columns show the latencies of first participants at Syracuse (NY), Florida State (FL) and Cardiff (UK), respectively. The next four columns show the latencies of the last participants in these sites. The last column shows the total amount of data transmitted to each site for each test case. Since the machine at Cardiff has 10 Mbps download capacity, we did not test it for more than 20 participants. Table 5-30 shows the loss rates and Table 5-31 shows the jitters for the same tests. All the columns are the same as Table 5-29.

Table 5-29 Latency values for single video meetings with one broker in distributed setting

Users per site	all users	Latencies of first users (ms)				Latencies of last users (ms)				BW/ps Mbps
		IN	NY	FL	UK	IN	NY	FL	UK	
4	16	1.52	13.29	11.1	65.64	1.7	13.62	12.02	67.77	1.2
10	40	2.68	13.95	13.49	69.7	3.29	14.75	14.43	75.51	3
20	80	3.84	15.71	13.17	74.27	5.21	17.33	14.84	84.17	6
50	150	5.79	18.96	17.11	N	8.47	22.66	20.63	N	15
100	300	10.55	25.93	25.82	N	16.1	33.48	32.44	N	30
200	600	22.26	44.51	42.45	N	33.54	59.45	55.23	N	60

There are two major factors that contribute to the latency values shown on the above table. These are the delays introduced by the broker to route the packages and the transmission time from the broker to the destination site. We tried to minimize the other factors as much as possible. For example, by running the transmitter client and the

broker at the same Linux cluster with gigabit connection between them, we tried to minimize the transmission time from the transmitting client to the broker. In addition, we tried to minimize the overhead on the receiver machine by running only four measuring receivers on one machine and running all other receivers at another machine in the same subnet. We should also note that the latency values seen at Indiana is mainly due to the routing delays by the broker, since these two sites are very close to each other and they have gigabit connection between them. Therefore, we can estimate the approximate transmission overheads for the remaining three sites by subtracting the routing overhead (the latency for Indiana site) from the total latency.

Since the number of participants is very low for the first test case, the routing overhead by the broker is very small. Therefore, the latency values at the first row show the minimum transmission times for packages to travel between the sites. The US sites have very low latency values, all are less than 14ms. That is excellent for real-time communication. Even the latency values for the clients at UK are less than 70ms, which is still very good for real-time communication.

When the number of clients at each site increases, both the routing overhead and the transmission latency increase. The routing overhead increases to 22ms for 600 participants as it can be seen from Indiana latency. The transmission overheads for both Florida and Syracuse only increase around 10ms from 4 participants to 200. Therefore, this test shows that even 200 video streams can be transferred between these US sites with very small increase on the transmission delays.

Table 5-30 Loss rates for single video meeting with one broker in distributed setting

Users per site	All users	Loss rates of first users (%)				Loss rates of last users (%)				BW/ps Mbps
		IN	NY	FL	UK	IN	NY	FL	UK	
4	16	0	0	0.04	0.25	0	0	0	0.29	1.2
10	40	0	0.02	0.04	4.01	0	0.05	0	5.43	3
20	80	0	0.04	0	14.81	0	0.02	0.05	29.73	6
50	150	0	0.02	0.07	N	0	0.04	0.02	N	15
100	300	0	0.02	0.25	N	0	0.02	0.27	N	30
200	600	0	0.56	0.53	N	0	0.67	0.56	N	60

Table 5-31 Jitter values for single video meeting with one broker in distributed setting

users per site	All users	Jitters of first users (ms)				Jitters of last users (ms)				BW/ps Mbps
		IN	NY	FL	UK	IN	NY	FL	UK	
4	16	0.44	0.54	0.86	3.77	0.48	0.59	1.81	3.41	1.2
10	40	0.97	1.25	1.46	9.85	0.94	1	1.25	8.28	3
20	80	1.92	2.65	2.54	18.92	1.81	1.88	1.8	14.03	6
50	150	3.71	5.71	5.35	N	3.51	4.26	3.48	N	15
100	300	7.7	11.61	10.82	N	7.36	8.79	7.14	N	30
200	600	15.63	22.96	21.2	N	14.99	17.4	14.22	N	60

When the number of users is increased to 10 in the second test case, the clients at UK experience more losses. Although the latency and jitter values are still in very good range, the loss rate becomes too high for a quality communication. The first receiver experiences a loss rate of %4.0 and the last user experiences a loss rate of %5.4. These loss rates are not acceptable for a quality communication. This shows that

although Cardiff had 10 Mbps download bandwidth, we can only utilize a fraction of this bandwidth for videoconferencing.

Since the jitter is the amount of variation on the latency of consecutive packages in a stream, the factors that cause the changes in the latency are the causes of the variations on the jitter. Therefore, the two main reasons for the jitter is the routing overhead by the broker and the transmission delays between the sites. Similar to the analysis of latency, we can say that the jitter experienced by the clients at Indiana is mainly due to the routing of the broker. We can calculate the approximate jitter caused by the transmission by subtracting the jitter of Indiana clients from the jitters of other sites. This analysis shows that the jitter caused by the broker is more dominant than the jitter caused by the transmission. This test shows that very high number of video streams can be exchanged between the sites in US with very small amount of jitter.

5.6.2 Video Meeting Tests with Multiple Brokers

We repeated the previous single video meeting test by running a broker in every site and connecting the local clients to the brokers in those sites. Everything else was the same. Four brokers were running in total. Three brokers were directly connected to the one at CS site in Indiana. The CGL site at Indiana did not run a broker, since it was very close to the other broker in Indiana. Therefore, all clients at CGL connected to the broker at CS site.

In this setting, one video stream is transferred between the sites independent of the number of the clients at each site except the sites in Indiana. The brokers at remote sites received the video stream from the broker at CS site. Remote brokers delivered this stream to their clients locally.

We had access to only one machine at Cardiff, and run both the broker and all receivers on the same machine. That was a Linux machine with 1 GHz Dual Pentium III processors and 1.5 GB of memory. We had access to two Sun machines at Syracuse. The more powerful one had 1 GHz Quad CPUs and 8GB of memory and the other had Dual 300 MHz CPUs and 1GB RAM. We ran the broker and the measuring receivers at the more powerful machine and the rest of the clients on the other machine. We had access to more than two machines at the other three sites. Therefore, we ran the broker, the measuring receivers and passive receivers at separate machines. Florida had multiple Linux machines that had 733 MHz Dual Pentium III processors and 512 MB of memory. As we pointed out in the previous sections, CGL site had 8 identical Linux machines with 2.4GHz Dual Intel Xeon CPU and 2 GB of memory. CS site also had 8 Linux machines with 2.8 GHz Dual Intel Xeon CPU and 2GB of memory. We should note that the machines at three remote sites had less computing power compared to the ones at Indiana. Moreover, the machines at these three remote sites were shared platforms and we did not have exclusive access rights. So, there were other processes running during our tests.

Table 5-32 shows the latency results of these tests. All the columns are the same as the ones at Table 5-29. Only the last column is missing, since the amount of the data transferred between the sites is the same during these tests. Contrary to the previous test, in this case the main broker at Indiana introduces very small amount of overhead. It delivers a copy of all packages to only three remote sites, in addition to the local clients. Since the delivery of remote brokers has priority over the local clients, very little overhead is introduced by the delivery of packages to local clients. Therefore, in

this case, the latency is introduced mainly by the transmission of packages between the sites and by the broker at the remote sites.

Table 5-32 Latency values for single video meeting with multiple brokers in distributed setting

users per site	all users	Latencies of first participants (ms)				Latencies of last participants (ms)			
		IN	NY	FL	UK	IN	NY	FL	UK
4	16	1.03	13.37	13.39	60.98	1.37	13.58	14.08	61.05
10	40	1.24	13.67	13.67	67.14	1.74	14.14	14.59	68.5
20	80	1.64	14.29	15.17	65.79	2.14	15.22	16.68	68.55
50	200	2.55	16.55	18.52	74.4	3.65	18.87	21.96	81.8
100	400	4.16	20.42	24.42	121.74	6.22	25.01	31.3	135.86
200	800	7.19	29.3	41.96	27640.91	11.24	38.65	55.7	27669.64

In this test, the transmission latency of packages between the sites should be very similar for all tests, since there is only one stream delivered for all test cases. This value should be slightly less than the total latency of the first receivers of the first test with 4 receivers on each site, since the other overheads are very small. Additional increases on the latency values are the result of the overhead introduced by the routing of the remote brokers. Therefore, in this case, the capacity of the remote machines is very important, particularly when the number of clients increases on a site. Since the machine at Indiana has much better computing power than others, it introduces the smallest amount of overhead to route even 200 clients (around 12ms). The brokers at Syracuse and Florida add more overhead but still can support 200 clients with good quality. The broker at Cardiff can support 100 clients but can not deliver the stream to 200 receivers.

Table 5-33 Loss rates for single video meeting with multiple brokers in distributed setting

users per site	all users	Loss rates of first participants (%)				Loss rates of last participants (%)			
		IN	NY	FL	UK	IN	NY	FL	UK
4	16	0	0	0	0	0	0	0.02	0
10	40	0	0	0.04	0	0	0	0.02	0
20	80	0	0.02	0.04	0	0	0.02	0	0
50	200	0	0.02	0	0	0	0.02	0	0
100	400	0	0.02	0.02	0	0	0.02	0.02	0
200	800	0	0	0.05	0	0	0	0.02	0

Table 5-34 Jitter values for single video meeting with multiple brokers in distributed setting

users per site	all users	Jitters of first participants (ms)				Jitters of last participants (ms)			
		IN	NY	FL	UK	IN	NY	FL	UK
4	16	0.43	0.75	1.62	1.95	0.79	0.71	2.39	1.76
10	40	0.48	1.02	1.5	2.56	0.77	1.01	1.52	2.3
20	80	0.73	1.38	2.36	4.34	0.69	1.3	1.91	3.9
50	200	1.44	3.4	5.36	10.8	1.26	3.14	4.28	10.23
100	400	2.78	6.45	10.48	20.16	2.53	6.08	8.5	19.35
200	800	5.53	13.13	20.75	32.83	5.14	12.42	17.27	31.16

Table 5-33 shows the loss rates for this test. Since the package losses occur mainly when packages are traveling between the sites, the loss rates for this test case is very small and independent of the number of participants in a site. Only one copy of the stream is traveling between the sites in all test cases. This reduces the package losses significantly and also eliminates the bandwidth limitations of transferring multiple

streams. For example, now Cardiff site can have as many receivers as their broker can support. They are not bounded by the network limitations.

Table 5-34 has the jitter values for this test. These jitter values are mainly introduced by the routing of remote brokers. Therefore, the increase on the jitter mainly depends on the computing power of the remote machines. The values at the first row show the maximum jitter introduced by the transmission of packages between the sites. The jitter introduced by the transmission is very small even for Cardiff. Therefore, this is another benefit of running a broker in a geographically distant site. In addition to bandwidth savings, running a broker at a distant site can also reduce the jitter experienced by the clients because of the transmission of multiple streams.

5.6.3 Video Meeting Tests with Better Machines at Cardiff

We have repeated the previous video meeting tests with multiple machines at Cardiff that have higher computing power and better network bandwidth. We repeated both the single and multiple broker tests. We had access to two machines at Cardiff that had 100 Mbps upload and download bandwidth. They had gigabit bandwidth between themselves. One of them was a Linux machine with 1.5 GHz Quad Intel Itanium 2 processors and 8.2GB of memory. The other was also a Linux machine with 1.2GHz Dual AMD Opteron processors and 2GB of memory. For the single broker tests, we used both machines to run the clients and for the multiple broker tests, we used the first machine to run the broker and the second machine to run the clients. All other machines and the settings were the same as the previous tests.

For this test, we only report the results for Cardiff site, since the other aspects of the test is the same as the previous case. Table 5-35 shows the latency, jitter and loss

rates for the video meeting tests with one broker at Indiana. In this test, Cardiff site was able to run clients for all test cases. Previously, Cardiff was able to run only 20 clients because of the bandwidth limitations. Therefore, in this test, the total number of participants on the broker is higher than the previous case, when there are more than 20 participants per site. This requires the broker to introduce more delays and jitters to video packages compared to the previous case.

Table 5-35 Test results of UK clients for a video meeting on one broker with better machines

users per site	all users	first user latency	last user latency	first user loss rate	last user loss rate	first user jitter	last user jitter
4	16	55.6	56.36	0.02	0	0.55	1.55
10	40	56.28	56.96	0	0	1.02	0.86
20	80	57.67	59.15	0	0	2.17	1.74
50	200	61.76	65.55	0	0	5.74	4.52
100	400	68.53	76.15	0.02	0	11.06	9.43
150	600	75.81	86.98	0	0	15.52	14.81
200	800	91.71	106.68	0	0	20.75	19.91

In this case, the performance of clients at Cardiff is very similar to the performance of the clients at the other two US sites in the previous test. The main difference is the initial high transmission latency for the clients at Cardiff. As the number of users per site increases, the latency and the jitter also increase for all sites. This increase for the clients at Cardiff is very similar to the increase experienced by the clients at the other two US sites. In addition, the loss rates for the clients at Cardiff are almost zero similar to the clients at the other two US sites. Therefore, this test shows

that 200 video streams can be transferred with excellent quality from Indiana to UK over the Atlantic. This is very important for videoconferencing applications and for our proposed architecture, since it shows that a media delivery network can provide excellent quality service with very high number of real-time streams even between the US and Europe. More tests for longer periods of times need to be done to evaluate the performance of these networks, but still it is a very good indicator.

Table 5-36 Test results of UK clients for a video meeting on multiple brokers with better machines

Users per site	all users	first user latency	last user latency	first user loss rate	last user loss rate	first use jitter	last user jitter
4	16	56.53	56.66	0	0	2.17	2.16
10	40	56.24	56.55	0	0	1.14	1.09
20	80	56.55	57.11	0	0	1.21	1.12
50	200	58.01	59.45	0	0	2.25	2.03
100	400	60.26	63.06	0	0	4.21	3.7
200	800	65.28	70.98	0	0	8.3	7.55

Table 5-36 shows the summary of the multi broker tests. As we can see, now the broker at Cardiff can easily deliver 200 video streams to its clients. While the minimum latency is 56.5 ms for the first client in the meeting with 4 users, the maximum latency for the last client in the meeting with 200 users is only 70.9ms. Therefore, the amount of overhead introduced by the routing at the broker is around 15ms for the worst case. This means that this broker can support much more clients. In

addition, now the jitter increases very slowly for Cardiff clients by providing much better service. It only increases 6ms from the minimum value.

We derive many observations from these tests. First one is the importance of running brokers at the right locations on the network. For example, when we ran a broker at Florida State, we realized that it could send out only 15 video streams at a time to outside world, since it had a maximum upload bandwidth of 5 Mbps. However, in the case of Indiana sites, the brokers can send out more than 800 video streams, since they have gigabit upload speeds. Therefore, it is very important to run brokers at high bandwidth locations.

As we expected, our results indicate that by running multiple brokers in geographically distributed sites, both the scalability of the system and the quality of the service provided can be increased significantly. In addition, by running a broker in a remote site, significant bandwidth savings can be achieved and the bandwidth limitations can be overcome to support more participants. While only a few clients were supported in Cardiff without running a broker over there, the number of clients increased to more than a hundred when we ran a broker.

Another important observation about the results of Cardiff clients is the benefit of running brokers at geographically distant locations. Since the transmission times are significantly higher for these sites, it is very important to run brokers at geographically distant sites and minimize the transmission delays. For example, if the publisher was in UK for the single broker test, then the transmission overhead of the streams to clients at UK would have been increased by two fold. The video stream would have traveled through the Atlantic ocean two times to reach to the clients at UK. Therefore, it is

critical to run brokers at geographically distant sites. In addition, these results show the importance of connecting the clients with correct brokers. For example, it is very important for clients at UK to be associated with the brokers over there. Otherwise, the quality of their communication would have been reduced considerably. When choosing a broker to connect, a client needs to take into account at least three important factors. These are the proximity of a broker to the client, the quality and the bandwidth of the network connection between the client and the broker, and the capacity and the load of the broker. By evaluating these factors, it can choose the best broker to connect.

Maybe the most important result of the wide area tests is the fact that the networks that we tested provided very good quality communication for audio/video streams. When transferring even very high number of video streams, they provided excellent service for real-time videoconferencing applications. The lost rates were very small and they can be negligible even for 200 video stream transfers. Similarly, the amount of latency and the jitter is very small. Even going through the Atlantic Ocean does not introduce a challenge. Therefore, the underlying network infrastructure is good enough to implement a distributed brokering system on top of it to deliver audio/video streams.

Chapter 6

Meeting Management Architecture and Services

In this chapter, we present service oriented architecture for managing videoconferencing sessions using a publish/subscribe content delivery middleware. First, we give an overview of different videoconferencing practices and explore the ways to implement these videoconferencing techniques using a publish/subscribe system. Then, we cover the details of service oriented architecture.

6.1 Overview of Online Meetings

Similar to real life meetings, there are many types of online meetings. We can classify these online meetings into three major categories; broadcast meetings, free discussion meetings, and moderated meetings.

Broadcast meetings are the simplest form of online multiparty meetings. We define it as a one speaker session. That is, audio and video communications are one way. Only the speaker has the right to send audio and video in a meeting. Participants

listen the speaker and watch the video. They may ask questions or make comments by using some other means than audio or video, such as chat or email. It can be used to teach an online class, or deliver a ceremony to a large number of people, or air a televised show over the Internet.

Free discussion meetings are usually for small number of participants and less structured. Participants generally know each other. Although there might be a discussion leader to facilitate the discussion, any participant can speak at anytime by observing the general rules of courtesy. AccessGrid meetings are a good example for free discussion meetings. Usually many groups of researchers at different locations interact with each other through AccessGrid online videoconferencing system. There is no moderator, and anybody can speak at any time. In addition, many small scale business meetings can be considered as free discussion meetings. For example, many small scale H.323 meetings are operated in this mode.

Moderated meetings are similar to real world panels, conferences or interviews. In moderated meetings, each participant has a role. We can categorize the most frequently used roles as listeners, speakers and moderators. Listeners participate in a meeting only by listening. They can ask a question or make a comment as long as they are granted the permission. They are like students in a classroom or participants in a conference. Speakers have the right to speak and they might talk anytime. There can be one or more speakers in a session. Listeners can be promoted to the speaker role temporarily when they are granted the microphone. Moderators are in charge of controlling meetings. They might have different responsibilities according to various meeting implementations, but usually they are in charge of accepting and rejecting

people into meetings, and granting or denying accesses to some shared resources such as microphones. In some meetings, one participant can be both a speaker and a moderator.

Broadcast meetings can be considered as a special case of moderated meetings in which there is only one speaker and many listeners. Free discussion meetings can also be considered as a special case of moderated meetings, in which all participants are speakers.

6.2 Meeting Implementation Techniques

6.2.1 Broadcast Meetings

In broadcast meetings, communication is one-to-many. Only the speaker can send audio and video streams to participants. There is one audio stream and one video stream in a meeting. Therefore, there is no need for audio or video mixing. This type of meetings can be implemented by publishing the audio stream to one topic and the video stream to another topic on the broker network. Participants subscribe to these topics to receive audio and video streams. Some participants might choose to receive only the audio or the video stream, in that case they subscribe only to that topic.

These meetings scale very well, since there is only one audio and one video streams in a meeting. Depending on the number of available brokers, thousands of users can be easily supported. In addition, since listeners don't send any audio or video streams, it is much easier to develop and deploy client applications. It would even be possible to use some widely known media players as clients. This would simplify joining meetings considerably by eliminating software downloads and installation.

In some meetings, multiple versions of the audio and video streams of the speaker can be transmitted to support users with different capabilities. While high end users can be offered good quality audio and video streams, low end users can be supported by less demanding codec types. In this case, more topics should be allocated for different audio and video encodings, and transcoders should be introduced to change the format of the audio and video streams published by the speaker.

6.2.2 Free Discussion Meetings

Similar to AccessGrid meetings, free discussion meetings are designed for high end users. Usually there are a small number of participants in a meeting. Each participant can receive and process multiple audio and video streams at the same time. In addition, almost all users transmit audio and video streams in a meeting.

The implementation of these meetings is very similar to the implementation of broadcast meetings. One topic is allocated for audio streams and another topic is allocated for video streams. All participants publish their audio streams to the audio topic and video streams to the video topic. All participants receive all the media streams in a meeting except their own streams. This eliminates the need to have media processing units at server side, such as audio and video mixing.

The main disadvantages of this solution are its scalability and its high demand of user capabilities. This solution demands more resources from broker network, since all media streams are delivered to all participants except the producers. In addition, since anybody can speak at anytime, when the number of participants increases, it becomes cumbersome to manage such an online meeting without any moderation functionality.

6.2.3 Moderated Meetings

In some moderated meetings, there can be multiple speakers. While in others there can be one speaker, but a listener can be promoted to the speaker role temporarily when asking a question or making a comment. In addition, moderated meetings should be able to support a diverse set of participants. Therefore, it should provide services for clients with various capabilities. High end clients can be supported by serving them multiple concurrent audio and video streams, and low end clients can be supported by serving them processed media streams according to the needs of those individuals. Furthermore, users should be able to choose the media streams they would like to receive and ignore the others in a meeting.

Contrary to the previous two meeting implementations, in this case, it is better to use multiple topic numbers for audio and video streams. The audio of each speaker should be published to a unique topic. Each video stream should also be published to a unique topic. Therefore, a participant can explicitly select the audio and video streams of his/her choice and subscribes to only those topics. In addition, media processing units can subscribe to some topics to receive the media streams, and publish back the processed media to another topic for users to access it.

The implementation of moderated meetings is more complicated than the previous two cases. First of all, a mechanism should be implemented to give users an option to choose the media streams they want. Secondly, a scalable media processing unit should be implemented. We provide the details on this architecture and the implementation on the following sections.

6.3 Moderated Meeting Architecture

In this section, we examine the implementation of moderated meetings. Broadcast and free discussion meetings can be implemented similarly. They can be considered as having a subset of components of moderated meeting architecture.

We design the architecture to provide scalability and fault tolerance. We also require it to be flexible enough to grow or shrink dynamically. It can run multiple instances of various service providers possibly in different locations to provide fault tolerance. Even though some components may fail, others continue to provide services without interruption. In addition, the architecture makes it easier to develop and maintain the various components of the system by having a clean separation of functions. This simplifies adding new services.

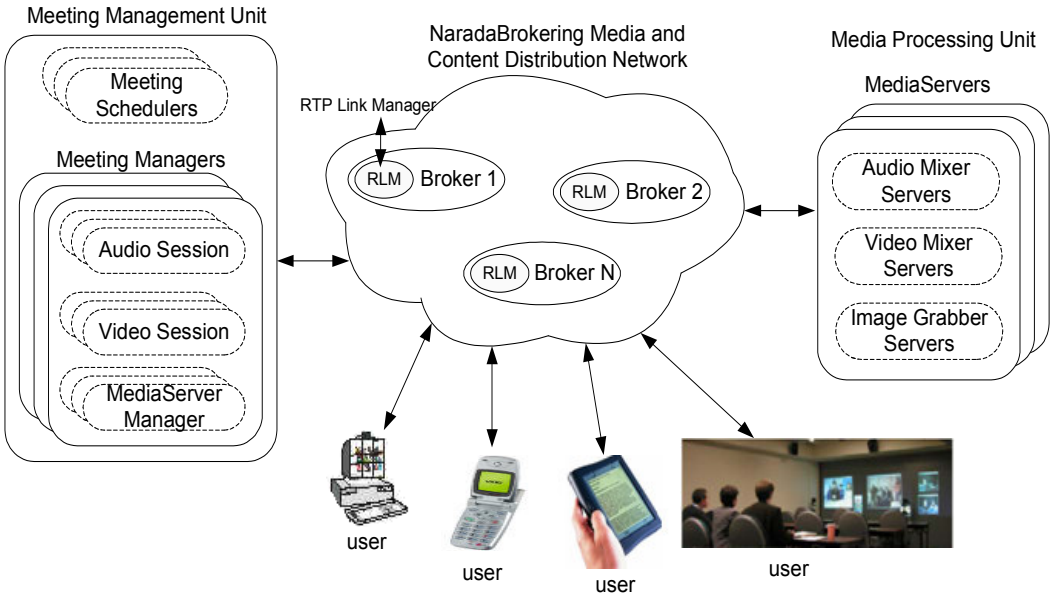

Figure 6-1 Moderated Meeting Architecture

We propose the following architecture (Figure 6-1) to meet these requirements. There are three main components of this architecture; Media and Content Distribution Network, Meeting Management Unit, and Media Processing Unit. When implementing broadcast and free discussion meetings, media processing units might not be provided, simplifying the system considerably.

Meeting Management Unit initiates and ends meetings. It also handles user joins and leaves. It has two main components; *MeetingSchedulers* and *MeetingManagers*. *MeetingSchedulers* create and delete sessions on *MeetingManagers*. *MeetingManagers* act as a factory for *AudioSession* and *VideoSession* instances. *MediaServerManager* is used by *AudioSession* and *VideoSession* instances to talk to media processing units. Each of these components can run multiple independent copies concurrently.

Media Processing Unit provides three different services; audio mixing, video mixing and image grabbing. Audio mixers combine multiple audio streams into one audio stream to support low end clients and save bandwidth. Similarly, video mixers merge four video streams into a single video stream to support low end clients. Image grabbers record the snapshots of video streams to provide users more information about them. More media processing services can easily be added by implementing the relevant interfaces and following the guidelines. All these media processing units can run multiple independent copies.

We provide a unified framework to manage the interactions among system components and distribute service providers. We use topics to identify the components in the system and provide them private communication channels.

6.3.1 Messaging Among System Components

We use reliable JMS messages to provide communications among various components in the system. This simplifies building a scalable solution, since messages can be delivered to multiple destinations without explicit knowledge of the publisher. Moreover, it provides location independence for each component, since a component is only connected to one broker and it exchanges all its data and media messages through this broker. In addition, using the same middleware for both data and media delivery reduces the overall system complexity considerably.

JMS provides a group communication medium. It uses topics as the group addresses. When a message is published on a topic, all subscribers of that topic receive that message. In our system, although some messages are intended to be sent to a group of destinations, many messages are destined to one target. Therefore, an efficient message exchange mechanism should be designed. Messages should only be delivered to intended destinations. In addition, topics should be organized in an orderly fashion. First, we examine various messaging types that take place in this system.

6.3.1.1 Messaging Semantics

There are three different messaging types in this system:

1. **Request/Response messaging**: This messaging semantic is used when an entity requests a service from a service provider in the system. It sends a request message to the service provider to execute a service and waits for a response message. The service provider processes the received message and sends a response message back to the sender. Since both the request and response messages are destined to one entity, it is important not to deliver these messages

to unrelated components. Therefore, all service providers and consumers should have unique topics to receive messages destined to them only. Most of the messaging in our system occurs in this fashion. All media processing units provide services to *AudioSession* and *VideoSession* components. *RTPLinkManagers* also provide services to *AudioSession* and *VideoSession* components. On the other hand, *AudioSession* and *VideoSession* components provide services to users.

2. **Group messaging**: This messaging semantic is used when an entity wants to send a message to a group of entities in the system. It publishes a message to a shared topic and all group members receive it. In some cases, receiving components send a response message back to the sender. In some other cases, no response message is assumed. There are two types of applications of this messaging semantic in our system. First one is to discover service providers. An entity sends a request message to the group address of some service providers. Then, each one of them sends a reply message including the information asked. Another application is to execute a service on a group of service providers. In this case, an entity sends a service execution request message to the group address, and all service providers in that group execute that service.
3. **Event based messaging**: Event based messaging is used when an entity wants to receive messages from another entity regarding the events happening on that component during a period of time, such as over the course of a meeting. All interested entities subscribe to the event topic name, and receive messages as the publisher posts them. A typical application of this event based messaging in

our system is to deliver events related to audio and video streams. All participants subscribe to the event topic and monitoring service publishes the events as they happen.

6.3.1.2 Topic Naming Conventions

To meet the requirements of the messaging semantics explained above, two types of topics are needed; group topics and unique component topics. We use a string based directory style topic naming convention to create topic names in an orderly and easy to understand fashion. All topic names start with a common root. We use our project name as the root name: GlobalMMCS. However, it is possible for an institution to change this root name and all topic names change accordingly. This allows installing more than one copy of this system on the same broker network. Group topic names are constructed by adding the component name to the root by separating with a forward slash. Groups are formed by the multiple instances of the same components. For example, all instances of MediaServers running in the system belong to the same group.

- GlobalMMCS/MeetingManager
- GlobalMMCS/AudioSession
- GlobalMMCS/VideoSession
- GlobalMMCS/AudioMixerServer
- GlobalMMCS/VideoMixerServer
- GlobalMMCS/ImageGrabberServer
- GlobalMMCS/RtpLinkManager

These strings are used as the component group addresses. For example, all AudioSession objects listen on GlobalMMCS/AudioSession topic to receive messages

which are intended to reach to all AudioSession objects. Similarly, all other objects listen on their group addresses to receive group messages.

Unique component topic names are constructed by adding the unique ids of each instance of that component to these component group addresses:

- GlobalMMCS/AudioSession/<sessionID>
- GlobalMMCS/VideoSession/<sessionID>
- GlobalMMCS/AudioMixerServer/<serverID>
- GlobalMMCS/VideoMixerServer/<serverID>
- GlobalMMCS/ImageGrabberServer/<serverID>
- GlobalMMCS/RtpLinkManager/<brokerID>

These unique topic names are used to communicate directly with a component. The messages sent to these topics only received by the component which has that id. When an instance of a component is initiated, it gets an id from the broker it is connected. Then it constructs its private topic name by following the above structure and starts listening on that topic for the messages destined to it.

Sometimes a component communicates with many different components; in that case, we use extra one more layer to distinguish these communication channels:

- GlobalMMCS/AudioSession/<sessionID>/RtpLinkManager
- GlobalMMCS/AudioSession/<sessionID>/AudioMixerServer
- GlobalMMCS/AudioSession/<sessionID>/RtpEventMonitor
- GlobalMMCS/VideoSession/<sessionID>/RtpLinkManager
- GlobalMMCS/VideoSession/<sessionID>/VideoMixerServer
- GlobalMMCS/VideoSession/<sessionID>/ImageGrabberServer

- GlobalMMCS/VideoSession/<sessionID>/RtpEventMonitor

In the above example, an AudioSession component communicates with three different entities: RtpLinkManager, AudioMixerServer and RtpEventMonitor. It uses different topics for each component. Using different topics simplifies logging and detecting the problems. It also simplifies developing codes to handle message exchanges with multiple components.

With this naming convention, we provide a unified mechanism to generate group and individual component topic names. It is easy to understand and debug.

6.3.2 Service Distribution Framework

In our system, we support multiple copies of the same service providers in a distributed fashion. Since, there are many types of service providers; we provide a unified framework (Figure 6-2) to distribute them. We assume that distributed copies should be able to run both in a local network and in geographically distant locations.

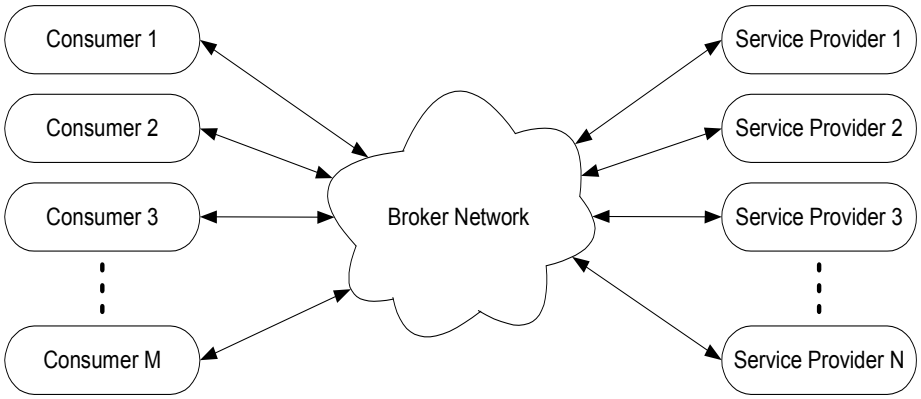

Figure 6-2 Service distribution framework

6.3.2.1 Addressing

Each service provider and consumer is identified by a unique topic name. This unique topic name is used to communicate with each entity privately. This topic name is generated as explained in the previous section. In addition to its own private topic, each service provider also listens on the service provider group topic to receive messages destined to all group members.

6.3.2.2 Service Discovery

Instead of using a centralized service registry for announcing and discovering services, we use a distributed dynamic mechanism. One problem with centralized registry is the failure susceptibility. Another difficulty is the fact that the status of the service providers change dynamically in our system. Therefore, it is not practical to update a centralized registry frequently.

In this approach, a consumer sends an *Inquiry* message to the service provider group address. It includes its own private topic name in this message, so that service providers can send the response messages back to it only. When service providers receive this message, they respond by sending a *ServiceDescription* message, in which they include their current status information and their private topic name. The status information depends on the nature of the service being provided. However, it must be helpful for the consumer to select the best service provider to ask for the service. The consumer waits for a period of time for responses to arrive, and evaluates the received messages. Since consumers do not know the current number of service providers in the system, after waiting for a while it assumes that it received responses from all service providers.

6.3.2.3 Service Selection

When a consumer receives *ServiceDescription* messages from service providers, it compares the service providers according to the service selection criteria set by user. This criteria can be as simple as checking the CPU loads on host machines and choosing the least loaded one or it can take into account more information and complicated logic. For example, users can be given an option to set the preferences over the geographical location of the service providers. This can be particularly useful for systems that are deployed worldwide.

6.3.2.4 Service Execution

Once the consumer selects the service provider on which it intends to run its service, it sends a *Request* message to the private topic of that service provider for the execution of the service. If the service provider can handle this request, it sends an *Ok* message as the response. Otherwise, it sends a *Fail* message. In the case of failure, the consumer either starts this process from the beginning or tries the second best option. A service can be terminated by the consumer by sending a *Stop* message.

In this system, a service is usually provided for a period of time, such as during a meeting. Therefore, the consumer and the service provider should be aware of each others continues existence during this time period. Each of them sends periodic *KeepAlive* messages to the other. If either of them fails to receive a number of *KeepAlive* messages, it assumes that the other party is dead. If the consumer is assumed dead, then the service provider deletes that service. If the service provider is assumed dead, then consumer looks for another alternative.

In this system, each service provider is totally independent of other service providers. Namely, service providers do not share any resources. Therefore, there is no need to coordinate the service providers among themselves. This simplifies the distribution and management of service providers.

6.3.2.5 Advantages of this Framework

Fault tolerance: There is no single point of failure in the system. Even though some components may fail, others continue to provide services.

Scalability: This model provides a scalable solution. There is no limit on the number of consumers to support as long as there are service providers to serve them. The fact that initially a consumer sends a message to all service providers, and they all respond back to the consumer, may limit the number of the supported service providers. However, this can be eliminated by limiting the number of service providers who respond to an *Inquiry* message. This selection can be based on the location of the service providers or some other criteria depending on the nature of the services provided. For example, already fully loaded service providers might ignore inquiry messages.

Location independence: All service providers are totally independent of other service providers and all consumers are also independent of other consumers. Therefore, a service provider or a consumer can run anywhere as long as they are connected to a broker.

6.3.3 Session Management

Since, audio and video streams travel independently from sources to destinations, audio and video sessions are managed separately, and their streams are processed independently. *AudioSession* objects manage audio sessions and *VideoSession* objects manage video sessions. *MeetingManager* objects act as factories for these session objects. They initialize and end them.

AudioSession and *VideoSession* objects have two main functions. First one is to manage the topics used for a meeting. They keep the list of users and the topics they publish their media. The second one is to provide session management services to participants, such as user joins and leaves. While handling these requests, they usually talk to other system components, such as media processing units and RTP link managers.

6.3.3.1 Session Distribution

Although, session management components are lightweight objects and they can handle large number of concurrent users, we still distribute these audio and video session objects to provide fault tolerance. We use the service distribution model outlined in the previous section. *MeetingManagers* act as service providers and *MeetingSchedulers* act as consumers. *MeetingSchedulers* can run either as an independent application or as an embedded application in a web server. When it is used with a web server, an administrator or a privileged user initiates meetings through a web browser.

MeetingSchedulers select the *MeetingManagers* based on their load. They initiate meetings on the manager that is serving the smallest number of participants.

Each meeting manager is completely independent of other meeting managers. Therefore, there is no need for a meeting manager to be in contact with others, or to cleanup resources after one fails.

6.3.4 Audio Session Management

Since audio communication is the fundamental part of a videoconferencing system, it is very important to provide best quality audio services. Transmission delays and package losses must be minimized. The broker network already minimizes the transmission delays by giving it the priority on routing.

Although, it is best to avoid audio processing at server side to eliminate losses and prevent additional delays, a number of factors make it unavoidable. First, some clients can receive only one audio stream. Some of them don't have enough bandwidth, and some don't have the capability of processing multiple streams. Therefore, audio mixing is necessary to provide a single mixed stream for those users, when there is more than one speaker in a session. Secondly, some clients may not support the audio encoding of a speaker, so transcoding may be necessary to encode in a different format.

Audio processing at server side adds transmission delays to packages for two reasons. First one is the extra transmission time. Instead of traveling from source clients to destinations directly, audio packages first travel to an audio processing unit, and then they travel to destination clients. The amount of this extra overhead depends on the location of audio processing units and the clients. If they are not located in geographically very distant places, it can be tolerable. For example, it takes around 32-45 ms for packages to travel from one coast to the other in US [MARKOPOULOU]. If clients and processing units are scattered around the US, in the worst case, this extra

travel would add around 90ms of latency. This extra overhead would still be acceptable. The second reason for the extra delay is the buffering and processing at the audio processing unit. Audio processing units implement a playout buffer algorithm [RAMJEE] to smooth out the jitters introduced by the transmission networks. This minimizes package losses. This buffering latency is usually similar to package sending interval times. Package sending interval times are mostly less than 60ms. The amount of latency introduced by audio processing is usually very small and can be negligible. It takes very small amount of overhead to process and send out audio packages. In summary, although audio processing at the server side may introduce significant delays, it can still be acceptable. Since the mouth-to-ear transmission times can be as much as 300ms for good quality and 400 ms for acceptable quality, the delays introduced by audio processing can still be tolerable.

The best way of implementing audio mixing in this architecture is to provide a common audio mixer for a session. This mixer receives the source streams from the broker network and publishes the mixed streams back on the broker network. Therefore, there is no direct link between the participants and the mixer. This has many benefits. First of all, it provides mixers location independence. They can be initiated anywhere as long as they are connected to a broker. Secondly, this solution provides scalability. Mixers handle the audio mixing part, and broker network handles the delivery part. Broker network can easily deliver mixed audio streams to thousands of participants. This would be very difficult to achieve in a centralized audio conferencing scenario, where all participants directly connected to a server, which mixes and serves the mixed streams to all clients. On the other hand, using one mixer may seem to be

problematic when supporting large number of speakers in a session. However, this is highly unlikely, because the real life meetings tend to have at most a dozen speakers in a session.

Another option might be to use an independent audio mixer to create each mixed audio stream. In this case, many audio mixers are created for a multiple speaker session. This approach has two main disadvantages. It consumes more bandwidth, since each mixer receives a copy of audio streams from the broker network. It also consumes more computing resources, since each audio stream is decoded and mixed multiple times in many mixers.

6.3.4.1 AudioSession Implementation

AudioSession component manages the audio part of a videoconferencing session. *AudioSession* objects reside in *MeetingManagers* and they are initialized and ended by them. When an *AudioSession* is initialized, it starts an *AudioMixerSession* at an *AudioMixerServer* to provide audio mixing services for this session. *AudioSession* component provides the following services:

- Add a speaker to an audio session
- Add a listener to an audio session
- Remove a participant from an audio session

Users join or leave this session by exchanging messages with this object. When an *AudioSession* object handles these actions, it exchanges messages with an *AudioMixerServer* and possibly many *RTPLinkManagers* inside NB brokers using the topic names shown on Figure 6-3. This communication is transparent to users. It may

add or remove users to/from an audio mixer session. It may also start and stop *RTPLinks* on *RTPLinkManagers* when needed to support legacy clients.

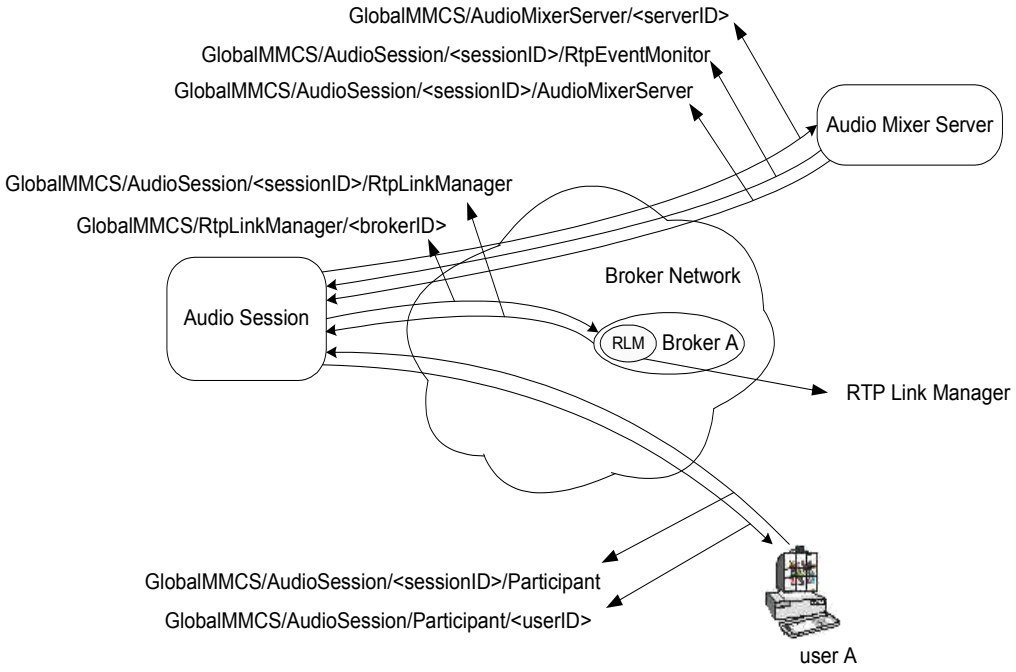

Figure 6-3 JMS message paths for an AudioSession instance

6.3.4.2 Creation and Deletion of an AudioSession

When an instance of *AudioSession* is created, it first communicates with *AudioMixerServers* to locate an available server to start an audio mixing session by using *MediaServerManager*. Then it creates an *AudioMixerSession* in the selected *AudioMixerServer* by exchanging JMS messages with it. When an *AudioSession* is deleted, it stops the *AudioMixerSession* in the *AudioMixerServer*.

6.3.4.3 User Joins and Leaves

AudioSession instances support two types of participants; speaker and listener. While speakers both have the right to send and receive audio streams, listeners can only receive the audio streams of the speakers. The audio stream of each speaker is published to one topic. Even if a user sends more than one streams, they are all published to the same topic. Therefore, each speaker is assigned one topic number to publish its audio stream. Since listeners don't publish any audio, they are not assigned any topic number.

When a listener joins a meeting, it can either subscribe to all audio streams in the meeting, or subscribe to the mixed audio stream of all speakers. If it subscribes to all audio streams, then it demands more bandwidth, but it directly receives the streams from the sources without any processing delay. If it chooses to receive the mixed audio stream, then it will receive one audio stream which is the combination of all audio streams in the meeting. In addition, if this client is a legacy client, an *RTPLink* is initiated for it in a broker.

When a speaker joins a session, in addition to above actions, the stream of this client is added to the audio mixer session. If it wants to receive a mixed audio stream, then a customized mixed audio stream is also created for it. This customized stream is published to another topic on the broker network and the speaker subscribes to that topic to receive it.

When a listener leaves the session, it unsubscribes from the topics it subscribed, and if an *RTPLink* was started for it, it will be deleted. When a speaker leaves, in

addition to the actions taken on the leave of a listener, it is removed from the audio mixer session.

6.3.4.4 Multicast Group Support

Supporting multicast groups in audio sessions is similar to supporting legacy clients. An *RTPLink* is started for a multicast group. This *RTPLink* receives the audio streams from the multicast users and publishes them on the topic provided. Although multicast groups usually have many streams from many users in one address, all streams are published to one topic. This is because users usually receive either all audio streams or none from a group.

6.3.4.5 Audio Mixing

Audio mixing is very common in videoconferencing systems. Since, it saves bandwidth by combining many streams into one, and requires relatively low computing power. Our implementation is similar to the one explained in [SINGH]. But it does not have the scalability problem mentioned there, since we use the broker network to deliver the mixed audio streams. It woks as in Figure 6-4. It provides an output stream for each speaker. It also provides one or more outputs for listeners. Although, the figure shows one output for listeners, more can be added when listeners require different encodings. In addition, some speakers might want to receive the audio streams of other speakers directly from the broker network. In that case, no mixed stream is constructed for those speakers.

When an *AudioSession* is initiated, it automatically starts an audio mixer. As speakers join the session, they are added to the mixer by the *AudioSession* object.

Audio mixer unit subscribes to the topic of a newly added speaker to receive its audio stream, and when the stream arrives, it constructs a processing line for that. When a speaker leaves the session, its processing line is removed from the mixer unit.

Figure 6-4 shows how audio mixing is performed. Decoders decode the received audio streams into a common linear format, (8 kHz sampling rate, 8bit per sample, mono). This is a telephone line quality audio. Then, repackatizers adjust the sizes of audio packages when necessary. Since we support a variety of clients, not all of them use the same package size. While Polycom client uses 60ms packages, Rat 4.2.2 uses 20ms packages. Currently we use 60ms as our systems package size. After repacking, packages wait in a queue to be picked up by the audio mixer. Audio mixer thread polls all queues regularly, and adds the values of all available data. Before the mixer, each sample of the voice is stored in a byte type (8bit), after mixing we keep each sample in a short type (16bit) to prevent any overflow or underflow. Then a copy of mixed audio data is passed to each subtracter. Subtracters subtract the data of themselves from the received mixed data if there is any, and store the result in a byte type, and pass it to the encoder. If the mixed audio sample value is out of range for byte type, the maximum or the minimum byte value is assigned accordingly. We have not experienced any distortion of audio because of this value casting. Encoders encode each stream according to the specific request of the user and publish them on the given topic number.

In some cases, a speaker may send multiple audio streams into a session. Particularly, multicast groups tend to send many streams, since they represent a group of participants instead of one endpoint. In this case, the handling of streams is the same

up to the mixer. After mixing, the subtracter subtracts the values of all streams belonging to that speaker from the mixed data. This prevents feeding back that users own audio to itself.

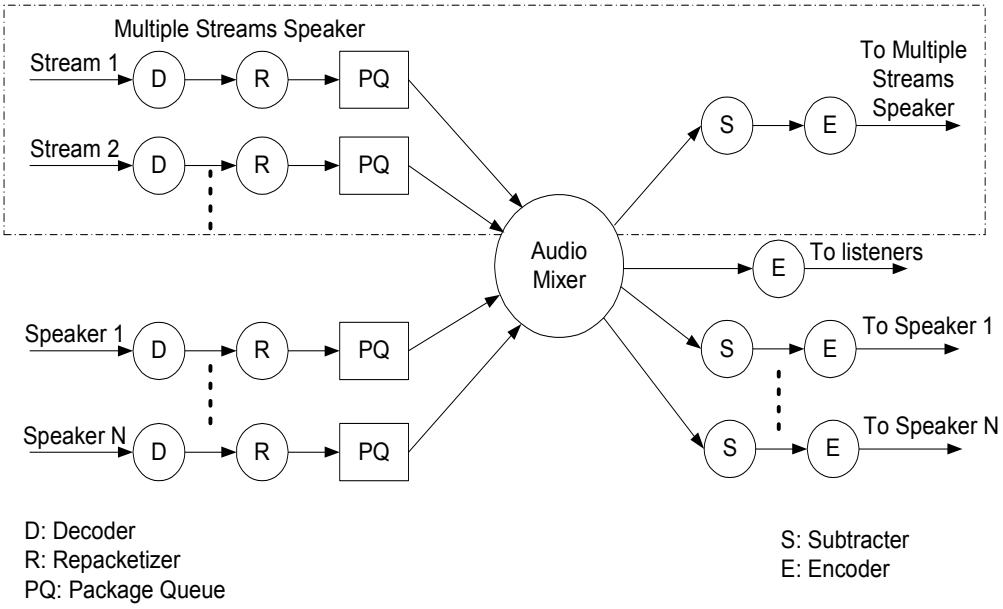

Figure 6-4 Audio mixing

6.3.4.6 Audio Mixing Performance Tests

While some audio codecs are computing intensive, some others are not. Therefore the computing resources needed for audio mixing change accordingly. Audio mixers need to have prompt access to CPU when they need to process received packages. Otherwise, some audio packages can be dropped and result in the breaks in audio communications. Therefore, the load on audio mixing machines should be kept at as low as possible.

Table 6-1 Audio mixer performance tests

Number of mixers	CPU usage %	Memory usage (MB)	Quality
5	12	36	No loss
10	24	55	No loss
15	34	73	No loss
20	46	93	Negligible loss

We have tested the performance of an *AudioMixerServer* for different number of mixers on it. There were 6 speakers in each mixer. Two of these speakers were continually talking and the rest of them were silent. There was also one more audio stream constructed with the mixed stream of all speakers. Therefore, 6 streams were coming into the mixer and 7 streams were going out. All streams were 64kbps ULAW. The machine that was hosting the mixer server was a WinXP machine with 512 MB memory and 2.5 GHz Intel Pentium 4 CPU.

Table 6-1 shows that this machine can support around 20 audio mixing sessions. But we should note that, in this test all streams are ULAW. This is not a computing intensive codec. When we had the same test with another more computing intensive codec, G.723, the same machine supported only 5 mixing sessions.

6.3.4.7 Audio Stream Monitoring

We implemented an audio stream monitoring service to monitor the status of audio streams in sessions. Users need to know the list of audio streams in a session and

their up to date status. They also learn about the identity of the speakers through this service. The monitoring service monitors the audio topic of each speaker in the session. It generates four types of events to indicate the changes on a stream:

- *NewStreamReceived*: This event shows that a user started sending an audio stream. Audio packages started arriving from that user.
- *ActiveToPassive*: This user does not send any packages for the last 1 minute. This user may have left the meeting or stopped sending audio.
- *PassiveToActive*: This user started sending data again, after becoming passive.
- *ByeEvent*: This user left the meeting. It either sent an RTCP bye package or *Leave* message to end the session.

We implemented the audio stream monitoring service embedded with the audio mixing service. Since the audio streams of all speakers in a session are received by audio mixers, we avoid receiving the same audio streams by audio stream monitors. In addition, audio stream monitoring service is not computing intensive; it only generates events and publishes them on the broker.

All stream events of a session are posted to a JMS topic. All users of this session subscribes to that topic to receive them.

6.3.5 Video Session Management

The requirements for the design of the video part of a videoconferencing system are significantly different than the requirements of audio part. Video streams require much more network bandwidth, and their processing requires much more computing power. On the other hand, contrary to audio sessions, participants in a video session do

not have to receive all video streams in a meeting. Some users might not receive even a single video stream in a meeting.

In video sessions, all video streams should be published on unique topics. Users should be able to select the video streams they want and they should not be delivered any extra video streams. Otherwise, unwanted video streams can consume their network resources and degrade the quality of their videoconferencing sessions.

Contrary to audio mixing, there is no easy way of mixing all video streams in a video session. However, it is still possible to mix a limited number of video streams into one. In our system, there are two types of video processing services: video mixing and image grabbing. More of them can be added. Video mixers merge four video streams into one and image grabbers record the snapshots of video streams regularly in a known image format. These images are delivered to users to help them make intelligent decisions about the video streams they want to receive.

Similar to audio mixers, video mixers and image grabbers receive the video streams from the broker network and publish the mixed video streams and the generated images back to the broker network. Therefore, they avoid having a direct communication link between the users and the processing units. This mechanism provides both location independence and scalability for processing units. They can be initiated anywhere as long as they are connected to a broker, and many instances of these servers can be initiated to support large number of processing requests.

6.3.5.1 VideoSession Implementation

VideoSession component manages the video part of a videoconferencing session. An instance of this object resides in an instance of a *MeetingManager* and it

can be initiated by an administrator using a *MeetingScheduler*. When an instance of *VideoSession* is constructed, it starts an *ImageGrabberSession* on an *ImageGrabberServer* to record the snapshots of the video streams in this session. *VideoSession* component provides the following services:

- Add a participant to the session
- Remove a participant from the session
- Create a video mixer for this session
- Delete a video mixer for this session
- Add a stream to a video mixer in this session
- Remove a stream from a video mixer in this session

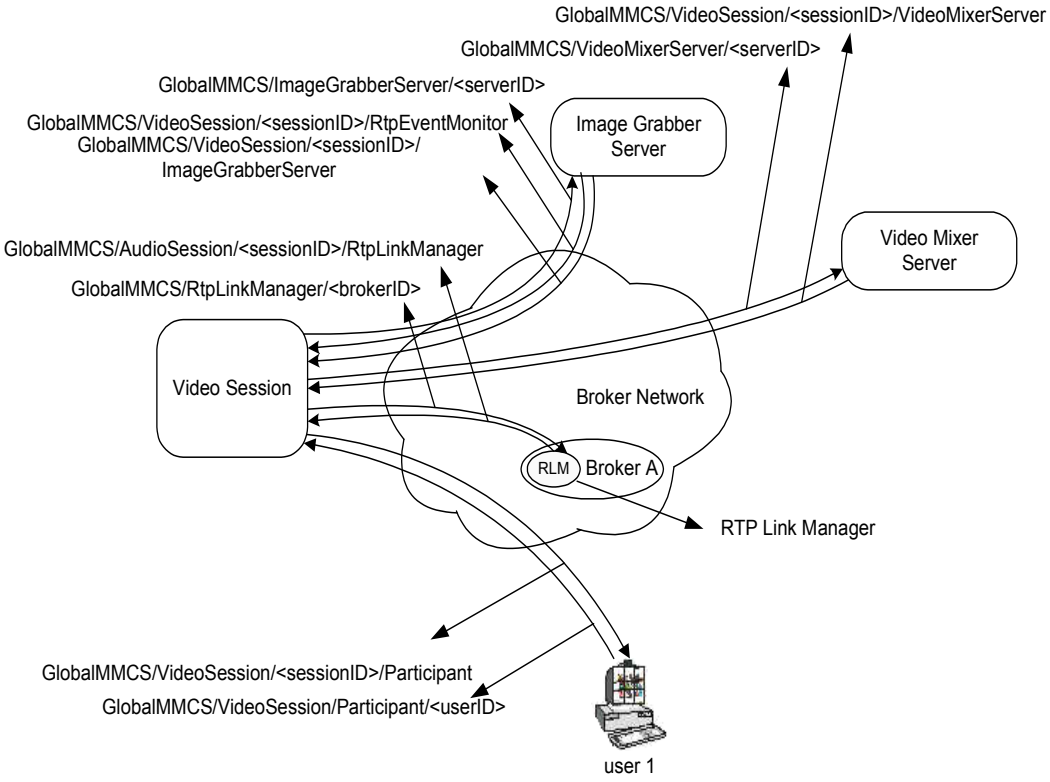

Figure 6-5 JMS message paths for a VideoSession

Users join or leave a video session by exchanging messages with a *VideoSession* object. In addition, administrators exchange messages with this object to create/delete video mixers and add/remove users. When a *VideoSession* object provides these services, it exchanges messages with an *ImageGrabberServer*, a *VideoMixerServer* and possibly many *RTPLinkManagers* inside NB brokers using the topic names shown on Figure 6-5.

6.3.5.2 Creation and Deletion of a VideoSession

When an instance of *VideoSession* is created, it first locates an available *ImageGrabberServer* to start an image grabbing session using a *MediaServerManager*. Then it creates an *ImageGrabberSession* in the selected *ImageGrabberServer* by exchanging JMS messages. When a *VideoSession* is deleted, it stops the *ImageGrabberSession* in that server.

6.3.5.3 User Joins and Leaves

There are two types of users in video sessions. While some users only receive video streams, some others both send and receive. When a sender client joins a session, a topic number is assigned for that user to publish its video stream. Currently, a unicast client is allowed to publish only one video stream. However, the messaging mechanism can easily be modified to allow them to publish more streams if needed.

When a client joins a meeting, it is given a list of available video streams in this meeting through a web page. Then, that user can select the streams from that list to view. In addition, if this client is a legacy client, an *RTPLink* is initiated for it in a

broker. If the joining client is sending a video stream, an image grabber is started to get the snapshots of its video stream in the image grabber server.

When a participant leaves the session, it unsubscribes from the topics it subscribed, and if an *RTPLink* was started for it, it will be deleted. If the leaving client is a sender, its image grabber is deleted too.

6.3.5.4 Multicast Group Support

Supporting multicast groups is a little different from supporting legacy unicast clients in video sessions. While unicast clients send one video stream, multicast groups tend to have many more streams. This requires the *MulticastRTPLink* to recognize different video streams and publish them on different topics. *MulticastRTPLink* identifies the packages belonging to the same stream by examining their SSRC numbers, since each RTP stream has a unique SSRC number in a meeting.

When a multicast group joins a session, a *MulticastRTPLink* is started on a multicast supporting broker and a *MulticastImageGrabber* is started on the *ImageGrabberServer*. This link is given a range of topic numbers (from n to n+m) to assign them for the received video streams. It assigns the topic numbers in increasing order. It assigns the first topic number n to the first received stream, and the second topic number n+2 to the second received stream, and so on. *MulticastImageGrabber* listens on these topics and generate stream events for this link.

6.3.5.5 Video Mixing

There are a number of ways to mix multiple video streams into one video stream. One option is to implement a picture-in-picture mechanism. One stream is

dedicated as the main stream and it is placed in the background of the full picture. Other streams are imposed over this stream in relatively small sizes. Another option is to place the main stream in a relatively larger area than other streams. For example, if the picture area is divided into 9 equal regions, main one can take 4 consecutive regions and remaining regions can be filled with other streams. In our case, we choose a simpler way of video mixing. We divide the picture area into four equal regions and place a video stream to each region.

A video mixer is added to a video session by an administrator or a privileged user by using a web interface. That user also selects the video streams to be added to the mixer from the web interface. It is also possible to automate mixer addition and stream management of a video mixer. Audio information of participants can be used to select the video streams to be mixed. When a video session is initiated, a video mixer can be started to serve this session. When there are four or less video streams, all available streams can be added to the mixer. When there are more than 4 video streams in a session, most active users can be identified by examining their audio contribution patterns. The video streams of the four most active users can be mixed. This list can be dynamically changed over the course of a session.

A video mixer unit receives the video streams from the broker network (Figure 6-6). It first decodes these streams into YUV format. This decoded image is stored in a buffer and continuously updated as packages arrive. A thread runs periodically to inform resizers to pickup the images from these buffers. Resizers reduce the sizes of the video streams to QCIF size, 176*144 pixels, if they are not already in that size. Many video streams used in video conferencing applications are CIF size, 352*288 pixels. So

they are reduced to one fourth of their original size. Video mixer places each of these four video streams to one corner of the whole picture constructed. Then the newly constructed stream is encoded and published to the broker network to be delivered to interested users. Figure 6-7 shows the mixed video streams in various media player windows.

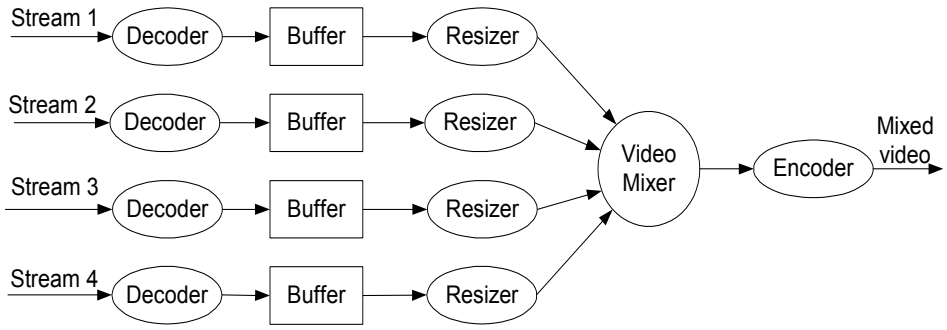

Figure 6-6 Video mixing

The frequency of the thread determines the frames produced per second in a video stream. When the number of frames per second (fps) increases, the quality of the video also increases by providing smoother video experience. 30 fps produce a very high quality video. For video conferencing applications, 10-15 fps can provide acceptable quality. On the other hand, higher frame rates consume more network bandwidth and CPU power.

Figure 6-7 Mixed video streams in various media players

6.3.5.6 Video Mixing Performance Test

Video mixing is a computing intensive process. One video mixer decodes four received video streams, and encodes one video stream as the output. Table 6-2 shows that a Linux machine with 1 GB memory and 1.8GHz Dual Intel Xeon CPU, can serve 3 video mixers comfortably and 4 at maximum. In this test, we used the same incoming video stream for all mixers. The incoming video stream was an H.261 stream with an average bandwidth of 150 kbps. The mixed video stream was an H.263 stream with 18fps.

Table 6-2 Video mixer performance tests

Number of Video mixers	CPU usage %	Memory usage (MB)
1	20	42
2	42	54
3	68	68
4	94	80

6.3.5.7 Image Grabbing

The purpose of image grabbing is to provide users with a meaningful video stream list in a session. Without the snapshots of the video streams, users are often confused to choose the right video stream for them. Snapshots provide a user friendly environment by helping them to make informed decisions about the video streams they want to receive. Therefore, it saves a lot of frustration and time by eliminating the need for trying multiple video streams before finding the right one.

As we have noted earlier, for each video stream an image grabber (Figure 6-8) is started. This image grabber subscribes to the video stream topic number to receive its video stream. Similar to video mixing, the stream is first decoded into YUV format, and then it is stored in a buffer. The image in this buffer is continuously updated by the decoder as the packages arrive from the broker network. Looping thread only picks the image from the buffer when it is time to generate a new image. When looping thread

picks up the image, first the resizer adjusts its size to QCIF size, 176*144 pixels, if it is not already in that size. Then encoder encodes the image in the JPEG format.

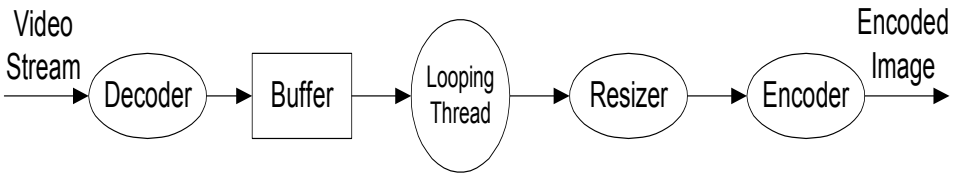

Figure 6-8 Image grabbing steps

When the image is generated, there are two options: One is to publish this image on a topic on the broker network and the other is to save it to a file. If it is published on the broker network, users access that image by subscribing to that topic. If it is saved to a file, users access it through a web server. The disadvantage of the second approach is its requirement for a web server. It requires a web server on each machine that is running an image grabber server. On the other hand, since the image is saved on a disk, it can be later accessed.

The most important variable when generating the snapshot images is the image generation interval. If images are generated too frequently, then image grabbing consumes more CPU time and also the images consume more network bandwidth to transfer to users. If they are saved very rarely, then the user might not have the up to date image of a stream. Although, the size of an image depends on the quality of the image and the content of the picture, on average it is around 5KB. We can calculate the bandwidth required from a user to receive the snapshot images in a session with the following equation (we assume that all images are equal size):

BW = Image Size * Image generation frequency * number of video streams

As the image generation frequency and the number of video streams increase, the bandwidth requirement also increases. If there are five video streams in a session and every 30 second an image is generated, then 12kbps is required to get these images regularly. For low bandwidth clients, instead of sending the pictures periodically, only a copy of them can be delivered when they join the session and for subsequent accesses their explicit request can be required.

Another important point is the even distribution of image saving times in a session. If all image grabbers save their images around the same time, then high network traffic will be generated on both the broker network and also on receiving endpoints. Therefore, it is important to distribute the image generation times in a session as even as possible.

When a stream is received it takes some time to receive packages for constructing a full picture. If the picture is saved before it is fully constructed, then users receive an incomplete picture. Therefore, before saving the first picture, it is better to check whether the picture is complete. The number of received packages or the number of bytes can be computed to predict the completeness of a picture.

6.3.5.8 Image Grabber Performance Tests

Image grabbing is also a computing intensive task. Each image grabbing includes decoding, resizing and encoding of a video stream. However, resizing and encoding are performed only when it is time to get the snapshot. Table 6-3 shows the performance tests for image grabbers. All image grabbers subscribed to the same video stream on a broker. That video stream was in H.261 format with an average bandwidth

of 150 kbps. Image grabbers saved a snapshot every 60sec to the disk in JPEG format. The host machine was a Linux machine with 1 GB memory and 1.8GHz Dual Intel Xeon CPU. These results show that 50 image grabbers can be supported on this machine.

Table 6-3 Image grabber performance tests

Number of Image grabbers	CPU usage %	Memory usage (MB)
10	15	66
20	35	110
30	50	148
40	60	192
50	70	232

6.3.5.9 Video Stream Monitoring

Similar to audio stream monitoring, we implemented a video stream monitoring service for video sessions. Video monitoring is particularly important for users, since they need to know the list of video streams in a session and their up to date status to be able to select the video streams they want. Video stream monitoring service monitors each video topic used in the session. It generates the same events as audio monitoring service: *NewStreamReceived*, *ActiveToPassive*, *PassiveToActive*, and *ByeEvent*.

We implemented the video stream monitoring service embedded with the image grabbing service. Since all video streams in a session are received by image grabbers,

we avoid receiving the same video streams by video stream monitors. All video stream events of a video session are posted to a JMS topic. All users of this session subscribes to this topic to receive these events.

6.3.6 Media Processing Service Distribution

Media processing framework (Figure 6-9) is designed to support addition and removal of new computing resources dynamically. A server container, *MediaServer*, runs in every machine that is dedicated for media processing. It acts as a factory for service providers. It starts and stops them. In addition, it advertises these service providers and reports the status information regarding the load on that machine. All service providers implement the interface required by the server container to be able to run inside. Each *MediaServer* is independent of other *MediaServers* and new ones can be added dynamically.

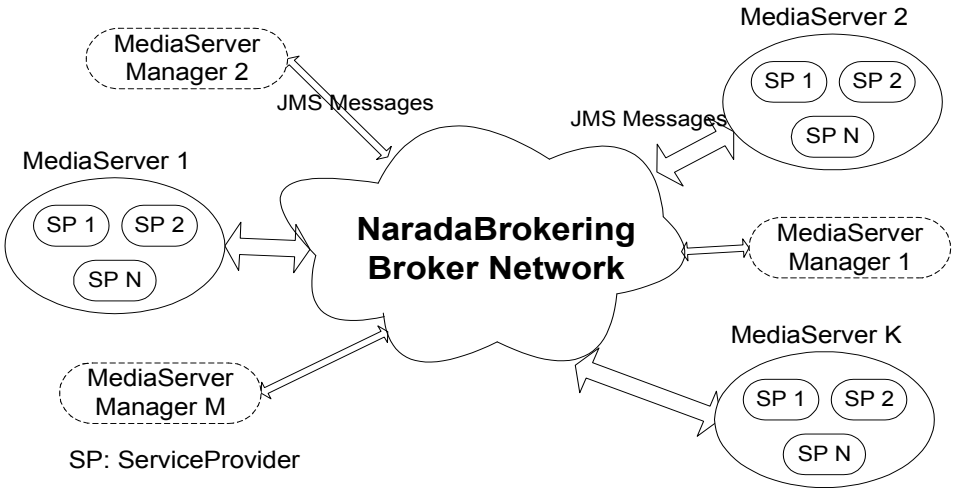

Figure 6-9 Media Processing Framework

Currently, there are three types of service providers for media processing: *AudioMixerServer*, *VideoMixerServer*, and *ImageGrabberServer*. More service providers can be added by following the guidelines and implementing the relevant interfaces. These service providers can either be started from command line when starting the service container, or they can be started by using the *MediaServerManager*. *MediaServerManager* implements the semantics to talk to *MediaServers*.

Media processing unit can be configured according to the needs of both small and large size organizations. For small organizations that will have only one or two concurrent meetings, one machine can be sufficient to run all media processing units. However, larger organizations need to run media processing servers on multiple machines. When distributing the servers, each machine should be dedicated to run one type of media processing service. It is particularly important for audio mixers to run on separate machines, since audio mixing is very sensitive and need prompt access to computing resources.

We use the previously explained service distribution model to distribute the media processing tasks. *MediaServerManager* implements the logic to talk to server containers and select the best available service providers. Currently, it selects the least loaded machines to initiate media processing services. More complicated algorithms can be developed to support higher number of processing units distributed around the world.

6.3.7 Broker Discovery and Selection

As we pointed out at the end of Chapter 5, it is very important for clients to connect to the right broker to get the best performance in distributed broker settings.

This process includes two steps: broker discovery and broker selection. First, users need to know the list of available brokers in the system, and then select the best one to connect.

There are two methods to discover the available brokers in a distributed broker network. In the first approach, the organization that is running the broker network keeps the list of all brokers. Users access this list either through a web site or sending queries with some known protocol such as web services. In the second approach, the broker network implements a dynamic broker discovery service. Users get the list of available brokers anytime by sending broker discovery request messages. Currently, there is no dynamic broker discovery service in NaradaBrokering. However, there is a project to implement such a service.

Once a user has the list of available brokers in the system, it can either select the closest broker manually or it can decide automatically. For example, in VRVS videoconferencing system, users are given the list of available brokers with their location through a webpage and they select the broker manually. Then, that user always connects to the broker network through that broker as long as it is using the same computer. When the broker is selected programmatically, users take into account three factors:

- The proximity of a broker to the client.
- The network bandwidth of the broker machine.
- The capacity and the load of a broker.

The user can calculate the proximity of brokers by calculating the transmission delays between the client and brokers. It can send/receive multiple messages to each

broker and calculates the round trip times. The bandwidth and the capacity of the broker machines can be given to users when the broker list is provided. Then, users choose the broker to connect with a heuristic function that takes into account these three factors.

Chapter 7

Conclusion

In this thesis, we presented scalable service oriented architecture for videoconferencing. We proposed using a publish/subscribe event brokering system for the delivery of audio and video streams in videoconferencing sessions. We used NaradaBrokering [NB1, NB3] publish/subscribe system to implement and demonstrate the ideas and concepts presented in this thesis. However, these ideas can also be incorporated into other publish/subscribe systems as long as they provide a scalable architecture. In addition to the changes that we incorporated into the NaradaBrokering publish/subscribe system; we conducted extensive performance tests to analyze the behavior of the broker network and to determine the capacity of it. We developed the quality assessment criteria to analyze the test results and to determine the quality of services provided. These tests demonstrated that this system can provide scalable videoconferencing services to very high number of participants in both single large size meetings and multiple smaller size meetings. In addition, the results of these tests provide guidelines

to organizations when deploying videoconferencing systems. These tests are very valuable since there is no available study in the literature that analyzes the behavior and the performance of software based videoconferencing systems.

We have also presented a novel architecture to manage videoconferencing sessions and distribute media processing service providers utilizing the reliable group communication services provided by the publish/subscribe middleware. This architecture provides a scalable service distribution framework that can grow or shrink dynamically. It supports additions and removal of computing resources without disturbing other system components. It provides dynamic service discovery and execution mechanisms in a distributed fashion without any central entity. Therefore, it provides a highly fault tolerant and robust framework to manage and distribute service providers. In addition, it avoids direct communication links among service consumers and service providers to provide location independence for both service consumers and providers. This results in a truly distributed system where service consumers and providers can be attached to the broker network at any point. We will summarize the answers of the research questions regarding this architecture in the following section.

7.1 The Summary of the Answers for the Research Questions

Here we summarize the answers of the 13 research questions presented at the Introduction chapter.

1. Is it possible to deliver real-time audio and video packages using an unreliable transport mechanism in publish/subscribe systems?

Since NaradaBrokering messaging network provides layered transport architecture [NB-TRANSPORT], it makes it easier to add new transport protocols. We implemented the *Link* interface provided by NaradaBrokering to incorporate UDP transport support. This made it possible for all audio and video traffic to be transported by UDP including on the links between clients and brokers, and brokers and brokers. Therefore, low latency package delivery services are provided end-to-end among the clients.

2. Is it possible to design a compact topic to be used for audio and video packages in videoconferencing sessions?

We designed an 8 byte long topic to be used as the communication channels among videoconferencing participants. The size of this topic is small enough to be added to all audio and video packages. We implemented a distributed topic generation mechanism to generate unique topics on space and time. Every broker runs a topic generator in the network and each topic generator generates unique topics when asked by clients. Each topic generator is independent of other topic generators and generates unique topics without interacting with any other topic generator in the network. UUID [UUID] is another similar unique id generation mechanism that generates ids for 16 bytes long and can not guarantee the uniqueness of the generated ids. Therefore, our solution is both efficient with smaller size and accurate with guaranteed uniqueness.

3. Is it possible to design a compact message type to encapsulate audio and video packages?

We designed a new compact event, *RTPEvent*, to carry audio and video packages in NaradaBrokering messaging network. We used the previously explained

compact topic in this event and modified the broker software to route this new event. This event has only four headers with minimal space requirements. The headers take 14 bytes when they are serialized to be sent over the network. Therefore, *RTPEvent* is a very compact message to carry audio and video packages in the broker network.

4. Can this system interoperate with other videoconferencing systems?

Currently almost all videoconferencing systems over Internet use RTP to carry audio and video traffic. There are two types of transport mechanisms for the delivery of RTP streams. The first one is the delivery of RTP streams over UDP and the second one is the delivery of RTP streams over IP-Multicast. We incorporated support for both RTP over UDP and RTP over Multicast solutions. We implemented *RTPLink* and *MulticastRTPLink* proxies in brokers to bridge legacy systems to the broker network. *RTPLinks* act as bridges between a broker and a legacy unicast client. *MulticastRTPLinks* act as bridges between multicast groups and the broker network. We also implemented an automatic RTP link management mechanism that starts RTP links automatically when legacy clients join sessions and removes them when they leave sessions. Therefore, our system interoperates with both multicast and unicast based videoconferencing systems.

5. What are the factors that affect the performance and the scalability of a broker and how does each factor affect it?

There are four factors that affect the performance and the scalability of a broker: audio/video package sizes, frequency of audio/video packages, the number of outgoing streams from a broker (the number of participants in a meeting), and the number of incoming streams to a broker (the number of meetings on a broker). The analysis of our

test results showed that the changes in audio/video package sizes do not affect the performance of a broker significantly. Since almost all audio and video packages are less than 1500 bytes long, the broker introduces very small amount of extra overhead for larger size packages on routing. Therefore, the dominant factor is the number of packages in a stream rather than the amount of data transmitted or the bandwidth of the stream. The frequency of packages in streams affects the performance of a broker significantly. Since, higher frequency streams have higher number of packages, they put more load on the broker. Our analysis showed that the number of participants or the number of outgoing streams affect the performance of a broker linearly. The broker spends equal amount of time to serve each participant and more participants put more load on the broker. When there are multiple meetings or multiple incoming streams to a broker, the broker resources are utilized better. The packages of these streams arrive randomly distributed on time, and the broker introduces smaller latencies to packages.

6. What is the capacity of a single broker in videoconferencing sessions?

We investigated the performance of a single broker in detail. We tested the performance for both single large size meetings and multiple smaller size meetings. In addition, we divided the tests into three different categories: audio only meetings, video only meetings, and audio and video combined meetings. Since the characteristics of audio and video streams are significantly different, it was necessary to test each of these three cases thoroughly.

The investigation of the performance of a single broker showed that a broker supports much higher number of participants with better quality services in audio only meetings compared to video only meetings. It supported 1500 participants in a single

audio meeting and it supported 400 participants in a single video meeting. Two main reasons for this are the uniform distribution of audio packages on time and the smaller number of packages in audio streams. In addition, these tests showed that large size video meetings can not fully utilize the broker resources because of the uneven distribution of video packages on time. Therefore, multiple smaller size video meetings utilized broker resources much better than the single large size video meetings. A broker can serve higher number of participants with better quality services in multiple smaller size video meetings. One broker supported 35 video meetings with 700 participants in total.

The single broker tests showed that a single broker can easily serve an organization with a few hundred videoconferencing users. It can support these users both in large scale meetings and multiple smaller size meetings. In addition, these tests help us predict the capacity of this system with other configurations. For example, a single broker would support higher number of participants on a better machine and smaller number of participants on a less powerful machine.

7. What is the impact of video stream delivery on audio stream delivery?

Audio and video combined meeting tests showed that the impact of video stream delivery on audio stream delivery is significant. Unevenly distributed video packages caused long delays on audio packages during the bursty video traffic. This reduced the quality of audio transmissions considerably. Therefore, we modified the routing algorithm at the broker to give priority to audio routing. This minimized the impact of video stream delivery on audio stream delivery and provided smooth audio

routing services. The new algorithm did not reduce the quality of video communications significantly.

8. What is the performance of the broker network in distributed settings with multiple brokers? Can it scale?

The investigation of the performance of the broker network revealed a weakness in the routing algorithm of NaradaBrokering that was limiting the scalability of the broker network significantly. The broker network was adding high overheads to packages that are traveling to other brokers in the system. This was limiting the number of brokers packages can travel. We modified the routing algorithm of the broker network to add minimum overhead to packages that are traveling to other brokers. This enabled packages to travel through multiple brokers. Large size meeting tests showed that the number of supported users can be increased linearly by adding new brokers. In addition, it showed that more brokers can be added to the broker network to provide better services to users with smaller latency, jitter and loss rate values. Multiple smaller size meeting tests showed that the scalability of the broker network can be increased significantly by adding new brokers as long as the users are distributed among the brokers properly.

9. Can this videoconferencing system be used among geographically distributed clients?

We conducted wide area tests with five different locations: Tallahassee (FL), Syracuse (NY), Cardiff (UK), and two sites in Bloomington (IN). These tests demonstrated that the broker network provides excellent middleware services to support clients in geographically distributed locations. The overhead introduced by the

transmission and the routing is very small. Even going through the Atlantic Ocean does not introduce a challenge. The quality of audio and video stream delivery among these geographically distributed clients was excellent. These tests also demonstrated that running brokers in distributed locations saves bandwidth and transmission times for audio/video packages. Therefore, it is very important for organizations with geographically distributed offices to run brokers in each site.

10. How will topics be used in videoconferencing sessions?

We analyzed various videoconferencing scenarios and identified three types of videoconferencing sessions: broadcast meetings, free discussion meetings, and moderated meetings. For the first two types of meetings, it is preferable to use common topics for all participants in a meeting. One pair of topics can be used for audio stream exchanges and another pair of topics can be used for video stream exchanges. However, in moderated meetings it is better to assign unique topics to each speaker. Each speaker publishes its audio and video streams on unique topics. This lets participants to choose the audio or video streams they prefer in a meeting and avoid receiving other streams in a session.

11. What kinds of components should there be in the system to manage the meetings? How will users start, discover and join meetings? What kinds of services will these components provide?

We designed a scalable and flexible meeting management framework. There are two components of this framework: *MeetingManager* and *MeetingScheduler*. *MeetingManager* is a container that has two components: *AudioSession* and *VideoSession.* It starts and ends audio and video sessions. *AudioSession* component

manages audio part of a videoconferencing session and *VideoSession* component manages video part of the session. They manage topic assignments to users and provide services to end users for joining and leaving sessions. *MeetingSchedulers* are used to discover, select and start audio and video sessions on *MeetingManagers*. This mechanism supports multiple instances of both *MeetingManagers* and *MeetingSchedulers.* Therefore, it provides a fault tolerant and scalable framework.

12. How will various components in the system communicate with each other? How will they utilize the underlying publish/subscribe system and use topics to interact with one another?

We designed a string based topic naming convention for all the components in the system. Each component is assigned a unique string topic. Therefore, each component can receive private messages destined to it. In addition, multiple copies of the same components are grouped together and assigned a group topic. A message published on the group topic is received by all group members. Private topics are mainly used to execute the services of a service provider and group topics are mainly used to discover the service providers in the system. This topic organization model provides an easy to understand and clear mechanism to manage communication channels among the systems components.

13. What kinds of media processing services are provided and how are they distributed among multiple available media processing units?

There are many types of audio and video processing services that can be supported in videoconferencing systems. We implemented three types of media processing services: audio mixing, video mixing and image grabbing. We have also

implemented a media stream monitoring service. Our media processing service framework supports additions/removal of processing services. It provides a scalable architecture to support high volume of media processing. It also allows the system to grow or shrink dynamically. There are many parameters when distributing media processing services and the distribution algorithms can change according to the available resources and the types of services provided. We have given a general algorithm to distribute media processing services. However, new algorithms can be developed and our framework makes it very easy to change the distribution algorithm.

7.2 Future Directions

As we pointed out at the end of the last chapter, more complete media processing service distribution algorithms need to be developed for large scale deployments. When there are multiple machines, or multiple clusters of machines in geographically distributed locations dedicated to run media processing services, the distribution of these service providers need to be done carefully to provide best services. These algorithms should take into account the required computing power and network bandwidth for the services. While some resources were idle, other services should not starve. In addition, it is important for some services to be as close as possible to participants in order to introduce minimum delays. Therefore, location information of the users and service providers can also be taken into account when distributing the services.

Although NaradaBrokering provides firewall, NAT and proxy traversal, we have not tested it extensively. The performance of going through these intermediary systems should not be a problem when UDP is supported; the performance of the media

delivery needs to be investigated when TCP is used. In some firewalls, only HTTP traffic is allowed. The delivery of audio and video streams over HTTP also needs to be investigated.

More performance and scalability tests may be conducted with higher number of brokers to observe the behavior of the broker network in larger scales. However, it is very difficult to arrange and perform such larger scale tests. First of all, it is very difficult to find the computing facilities to perform the tests. Secondly, it is very demanding to setup and collect results.

Bibliography

[3G] Dave Wisely, Philip Eardley, Louise Burness. IP for 3G: Networking Technologies for Mobile Communications. John Wiley & Sons (August 2, 2002). ISBN: 0471486973

[AG] The Access Grid Project. http://www.accessgrid.org/

[AG-SEC] Robert Olson, Certificate Management in AG 2.0, http://fl-cvs.mcs.anl.gov/viewcvs/viewcvs.cgi/AccessGrid/doc, March 5, 2003

[ALMEROTH] K. Almeroth, "The Evolution of Multicast: From the MBone to Inter-Domain Multicast to Internet2 Deployment", IEEE Network, Jan 2000, Volume 14.

[BALDONI] R. Baldoni, M. Contenti, A. Virgillito. The Evolution of Publish/Subscribe Communication Systems. "Future Directions of Distributed Computing", Springer Verlag LNCS Vol. 2584, 2003

[BANAVAR] G. Banavar, T. Chandra, R. Strom, and D. Sturman. A case for message oriented middleware. In Proceedings of the 13th International Symposium on Distributed Computing (DISC 99). 1–18.

[BAYEUX] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. Katz, and J. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In 11th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video, 2001.

[CALYAM] Prasad Calyam, Mukundan Sridharan, Weiping Mandrawa, Paul Schopis: Performance Measurement and Analysis of H.323 Traffic. PAM 2004: 137-146.

[CASNER] S. Casner and S Deering, "First IETF Internet Audiocast," ACM Comp. Commun. Rev., July 1992, pp. 92-97

[CLAYPOOL] M. Claypool and J. Tanner. "The Effects of Jitter on the Perceptual Quality of Video", In Proceedings of the ACM Multimedia Conference, Vol. 2, Orlando, Florida, USA, November 1999.

[COTTRELL] Les Cottrell, Warren Matthews and Connie Logg. Tutorial on Internet Monitoring & PingER at SLAC, http://www.slac.stanford.edu/comp/net/wan-mon/tutorial.html#loss

[CUSEEME1] Dorcey, T. CU-SeeMe Desktop Video Conferencing Software, in Connexions 9, 3 (March 1995)

[CUSEEME2] Jefferson Han and Brian C. Smith. CU-SeeMe VR: Immersive desktop teleconferencing. In ACM Multimedia, Boston, MA., 1996.

[ESM] Yang Hu Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In Proc. ACM SIG-METRICS Conference (SIGMETRICS '00), June 2000.

[EUGSTER] P. Th. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. ACM Computing Surveys, Vol. 35, No. 2, June 2003, pp. 114–131.

[FOX] Geoffrey Fox et al. "Grid Services For Earthquake Science". Concurrency & Computation: Practice and Experience. Special Issue on Grid Computing Environments. Volume 14:371-393.

[FRIEDMAN] Thomas L. Friedman. The World Is Flat: A Brief History of the Twenty-first Century. Publisher: Farrar, Straus and Giroux. ISBN: 0374292884. April 2005.

[FVC1] First Virtual Communications, http://www.fvc.com

[FVC2] Wainhouse Research, LLC, "Will Your Next Video Bridge Be Software-Based?", white paper, http://www.wainhouse.com/whitepapers/index.html, 2003.

[FVC3] First Virtual Communications, "Linking & Cascading Conference Servers", Technology White Paper, http://www.fvc.com, 2003

[G114] ITU-T Recommendation G.114, One Way Transmission Time. (05/2003).

[G711] ITU-T Recommendation G.711, "Pulse Code Modulation (PCM) of Voice Frequencies," 1988.

[GLOBUS-SEC] Implementation of the Globus Security Policy: v0.1, http://www.globus.org/security/implementation.html

[GMMCS] Global Multimedia Collaboration System. http://www.globalmmcs.org.

[GRYPHON] The Gryphon System, http://www.research.ibm.com/gryphon/index.html.

[GUNDUZ] Gurhan Gunduz, Shrideep Pallickara and Geoffrey Fox. A Framework for Aggregating Network Performance in Distributed Brokering Systems. Proceedings of the 9th International Conference on Computer, Communication and Control Technologies.Volume IV pp 57-63.

[H235] ITU-T Recommendation H.235, "Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals", 08/2003.

[H243] ITU-T Recommendation H.243, “Procedures for establishing communication between three or more audiovisual terminals using digital channels up to 1920 kbit/s”, Geneva, Switzerland, Feb. 2000.

[H263] ITU-T Recommendation H.263, “Video coding for low bit rate communication,” , 1998.

[H323] ITU-T Recommendation H.323, “Packet based multimedia communication systems”, Geneva, Switzerland, Feb. 1998.

[HANDLEY] M.Handley, J.Crowcroft, C.Bormann and J.Ott, "Very large conferences on the Internet: the Internet multimedia conferencing architecture", Computer Networks , pp.191-204, Volume 31, 1999.

[ISAACS] Isaacs, E. and Tang, J. “What video can and can't do for collaboration: A case study.” In Proceedings of ACM Multimedia '93. Anaheim: ACM, 1993.

[JMF] Sun Microsystems, Java Media Framework 2.1, http://java.sun.com/products/java-media/jmf/2.1.1/index.html, 2001

[JMS] Mark Happner, Rich Burridge and Rahul Sharma. Sun Microsystems. Java Message Service Specification. 2000. http://java.sun.com/products/jms

[MARKOPOULOU] A. P. Markopoulou, F. A. Tobagi, and M. J. Karam, "Assessing the Quality of Voice Communications Over Internet Backbones", IEEE/ACM Transactions on Networking, V. 11 N. 5, October 2003

[MSEC1] Shouhuai Xu, Ravi Sandhu, “Authenticated multicast immune to denial-of-service attack”, Proceedings of the 2002 ACM symposium on Applied Computing, Madrid, Spain.

[MSEC2] Refik Molva, Alain Pannetrat, "Scalable multicast security with dynamic recipient groups", ACM Transactions on Information and System Security (TISSEC), Volume 3 Issue 3, August 2000.

[NB1] http://www.naradabrokering.org

[NB2] Geoffrey Fox and Shrideep Pallickara, "The Narada Event Brokering System: Overview and Extensions". Proceedings of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02). CSREA Press (2002)

[NB3] Geoffrey Fox and Shrideep Pallickara, "NaradaBrokering: An Event Based Infrastructure for Building Scaleable Durable Peer-to-Peer Grids". Chapter 22 of Grid Computing: Making the Global Infrastructure a Reality Grid. Published by John Wiley, West Sussex, England. ISBN 0-470-85319-0. 2003.

[NBJMS] Geoffrey Fox and Shrideep Pallickara. "JMS Compliance in the Narada Event Brokering System". Proceedings of the International Conference on Internet Computing. June 2002. pp 391-402.

[NBNTP] Hasan Bulut, Shrideep Pallickara and Geoffrey Fox. Implementing a NTP-Based Time Service within a Distributed Brokering System. (To appear) ACM International Conference on the Principles and Practice of Programming in Java. June 16-18, 2003, Ireland.

[NBSEC] Shrideep Pallickara, Marlon Pierce, Geoffrey Fox, Yan Yan, Yi Huang, "A Security Framework for Distributed Brokering Systems", http://www.naradabrokering.org

[NBTRANSPORT] Shrideep Pallickara, Geoffrey Fox, John Yin, Gurhan Gunduz, Hongbin Liu, Ahmet Uyar, Mustafa Varank, “A Transport Framework for Distributed Brokering Systems”. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications. (PDPTA'03).Volume II pp 772-778.

[NTP] D.L. Mills. "Network Time Protocol (Version 3). Specification, Implementation and Analysis". IETF RFC 1305. (March 1992)

[OVERCAST] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and Jr. J. W. O’Toole. Overcast: Reliable multicasting with an overlay network. In Proc. of the 4th Symposium on Operating Systems Design and Implementation, 2000.

[POLYCOM] Polycom Inc. http://www.polycom.com

[RADVISION] Radvision Ltd. http://www.radvision.com

[RAMJEE] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks", in Proc. of IEEE Infocom 1994, Toronto, Canada, June 1994.

[RFC1191] J. Mogul, S. Deering, "Path MTU Discovery", IETF RFC 2736, Nov 1990.

[RFC2736] M. Handley, C. Perkins, "Guidelines for Writers of RTP Payload Format Specifications", IETF RFC 2736, Dec 1999.

[ROSENBARG] J. Rosenberg and H. Schulzrinne, "Models for multi party conferencing in SIP," Internet Draft, Internet Engineering Task Force, Nov. 2000. Work in progress.

[RTP] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. IETF RFC 3550. July 2003. http://www.ietf.org/rfc/rfc3550.txt.

[SIENA] A. Carzaniga, D. Roseblum, and A. Wolf. Design and evaluation of a wide-area event notification service. ACM Trans. Comput. Syst. 19, 3 (Aug. 2001), 332–383.

[SINGH] K. Singh, G. Nair, and H. Schulzrinne. "Centralized conferencing using SIP." In Internet Telephony Workshop 2001, New York, Apr. 2001.

[SIP] J. Rosenberg et al., "SIP: Session Initiation Protocol", RFC 3261, Internet Engineering Task Force, June 2002, http://www.ietf.org/rfc/rfc3261.txt

[SOA1] Kishore Channabasavaiah, Kerrie Holley, Edward M. Tuggle, Jr., Migrating to a service-oriented architecture, Part 1. Whitepaper. 16 Dec 2003. http://www-106.ibm.com/developerworks/library/ws-migratesoa

[SOA2] Kishore Channabasavaiah, Kerrie Holley, Edward M. Tuggle, Jr., Migrating to a service-oriented architecture, Part 2. Whitepaper. 16 Dec 2003. http://www-106.ibm.com/developerworks/library/ws-migratesoa2

[T120] ITU-T Recommendation T.120, "Data protocols for multimedia conferencing", July 96.

[TAPESTRY] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. Katz, and J. Kubiatowicz. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, University of California at Berkeley, Computer Science Division, April 2001.

[TOGA1] J. Toga and H. ElGebaly, “Demystifying multimedia conferencing over the Internet using the H.323 set of standards,” Intel Technology Journal, 2nd quarter 1998.

[TOGA2] J. Toga and J. Ott, “ITU-T standardization activities for interactive multimedia communications on packet-based networks: H.323 and related recommendations”, Computer Networks and ISDN Systems, Vol. 31, pp. 205–223, Feb. 1999.

[UUID] Paul J. Leach, Rich Salz. UUIDs and GUIDs. IETF Internet Draft. February 4, 1998.

[VC-CB] Videoconferencing Cookbook Version 3.0, Video Development Initiative, Advanced Videoconferencing Components and Management, http://www.videnet.gatech.edu/cookbook, April, 2002

[VRVS] Virtual Rooms VideoConferencing System. http://www.vrvs.org/

[VRVS2] David Adamczyk, et al, "Global Platform for Rich Media Conferencing and Collaboration", CHEP03 March 24-28, 2003 La Jolla, California

[VRVS-SEC] Kun Wei, "Videoconferencing Security in VRVS 3.0 and Future", ViDe 5th Workshop, March 25, 2003.

[WANG] Minjun Wang, Geoffrey Fox and Shrideep Pallickara, “A Demonstration of Collaborative Web Services and Peer-to-Peer Grids”, The proceedings of IEEE ITCC 2004 International Conference on Information Technology, Las Vegas.USA.

Glossary

BGMP (Border Gateway Multicast Protocol): BGMP is a protocol for inter-domain multicast routing. IETF RFC 3913.

G.711: G.711 is an audio compression and de-compression standard by the ITU. It is coming from telecommunications industry and it has been primarily used in telephony. There are two main algorithms defined in the protocol; ULAW and ALAW. ULAW is mostly used in US and ALAW is used in the rest of the world. Both algorithms are logarithmic. They require 64kbps of bandwidth.

G.723.1: G.723.1 is an audio coding standard by ITU. It is a dual rate speech coder with two bit rates, 5.3 kbps and 6.3 kbps. It is commonly used in videoconferencing sessions over Internet particularly when the bandwidth saving is important. It provides a telephone quality speech. The encoder encodes an audio package every 30ms.

H.225.0: H.225.0 has two major parts: Call signaling and RAS (Registration, Admission and Status). H.225.0 call control signaling is used to setup connections between H.323 endpoints (connecting, maintaining, and disconnecting calls). H.225.0/RAS is used to perform registration, admission control, bandwidth changes, status, and disengage procedures between endpoints and gatekeepers.

H.245: H.245 is a control signaling protocol in the H.323 multimedia communication architecture. It is used to exchange capabilities of endpoints and to open and close logical multimedia channels between the endpoints in a session.

H.261: H.261 is a widely used video coding standard by the ITU. It is originally designed for ISDN lines and encodes the data as the multiples of 64kbps. Today, RTP can be used to transport H.261 video stream over Internet and any transport mechanism can be used. H.261 encoder encodes most video frames with respect to another reference frame to remove temporal independence. However some frames are encoded with reference to itself to provide full picture updates. H261 supports two image resolutions, QCIF (Quarter Common Interchange format) which is (144x176 pixels) and CIF (Common Interchange format) which is (288x352 pixels).

H.263: H.263 is a video encoding standard by the ITU. It is evolved from the earlier H.261 standard and provides better compression. H.263 video streams can also be transported using RTP protocol over Internet. On average, H.263 requires half the bandwidth to achieve the same video quality as in the H.261. H.263 also provides more options for picture sizes. In addition to QCIF and CIF, it supports SQCIF, 4CIF, and 16CIF. SQCIF is approximately half the resolution of QCIF. 4CIF and 16CIF are 4 and 16 times the resolution of CIF respectively.

H.323: H.323 is a videoconferencing protocol for package based multimedia communications systems from International Telecommunications Union. It specifies real-time audio, video and data communications. It is an umbrella standard and includes many other standards such as H.225.0, H.245, T.120. The most recent version is H.323 version 5 (2003).

IGMP (Internet Group Management Protocol): IGMP is used by IP hosts to report their multicast group memberships to any immediately-neighboring multicast

routers. IGMP is an integral part of IP and it is required to be implemented by all hosts wishing to receive IP multicasts. IETF RFC 3376.

ISDN (Integrated Services Digital Network): An international communications standard for sending voice, video, and data over existing telephone lines by digitizing them. There are two types of ISDN. ISDN Basic Rate Interface (BRI) supports the total data rate of 144 Kbps. ISDN Primary Rate Interface (PRI) supports 1.5 Mbps in total. The basic service is intended for individual users and the primary service is intended for higher capacity requirements.

MADCAP (Multicast Address Dynamic Client Allocation Protocol): MADCAP is a protocol that allows hosts to request multicast address allocation services from multicast address allocation servers. IETF RFC 2730.

MSDP (Multicast Source Discovery Protocol): MSDP describes a mechanism to connect multiple IP Version 4 Protocol Independent Multicast Sparse-Mode (PIM-SM) domains together. IETF RFC 3618.

NAT (Network Address Translation): A NAT router acts as an agent between the public Internet and a private network. It allows many computers to connect to the Internet with one IP address. Many small companies and organizations use a NAT router to connect to the Internet. IETF RFC 3022.

PIM-DM (Protocol Independent Multicast – Dense Mode): PIM-DM is similar to PIM-SM. It specifies a protocol for routing multicast groups that may span wide-area (and inter-domain) internets. It is a less scalable protocol than PIM-SM, since it requires the state information be kept on all routers in a network. IETF RFC 3973.

PIM-SM (Protocol Independent Multicast – Sparse Mode**)**: PIM-SM specifies a protocol for efficiently routing multicast groups that may span wide-area (and inter-domain) internets. It is a more scalable protocol than PIM-DM, since it requires the state information be kept only in routers between the sender and receivers. IETF RFC 2362.

RAT (Robust Audio Tool): Similar to VIC, RAT is an audio conferencing tool that is used to send/receive and play audio streams. It is multicast capable and receive and play multiple audio streams simultaneously. It is an open source project and currently maintained by Network and Multimedia Research Group in University College London. http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/

RTP (A Transport Protocol for Real-Time Applications): RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. It is the most commonly used protocol to transfer audio and video streams over the Internet. IETF RFC 3550.

T.120: T.120 is data conferencing standard from ITU that provides real-time communication between two or more entities. It provides services such as application sharing, whiteboard sharing, file exchange, and chat. Although T.120 may also be used stand-alone, it is usually used with other protocol, such as H.323 and SIP.

UDP (User Datagram Protocol): The User Datagram Protocol is a transport protocol on top of IP layer. It provides connectionless best effort package delivery mechanism between two endpoints in Internet. It does not guarantee packages to be

delivered to the target address. It is commonly used to transfer audio and video traffic on Internet. IETF RFC 768.

UUID (Universal Unique IDentifier): UUID is a mechanism to provide unique ids in time and space for objects in distributed settings. Each UUID is a 128 bits (16bytes) number. UUID generators use hardware addresses (unique MAC addresses network cards), timestamps and random seeds to provide uniqueness. When MAC addresses are not available, UUID generators can not guarantee the uniqueness of generated ids. They rely on random number generators to minimize the possibility of collisions. UUIDs are also knows as GUIDs (Globally Unique IDentifier).

VIC (Videoconferencing Tool): VIC is a video conferencing tool that is used to send/receive and display video streams. It is commonly used in videoconferencing systems that use multicast such as AccessGrid. It is originally developed by Network Research Group at the Lawrence Berkeley National Laboratory in collaboration with the University of California, Berkeley. Currently, it is maintained by Network and Multimedia Research Group in University College London. http://www-mice.cs.ucl.ac.uk/multimedia/software/vic/

VNC (Virtual Network Computing): It is remote control software which allows a user to view and interact with one computer (the "server") using a simple program (the "viewer") on another computer anywhere on the Internet. It is an open project and can be downloaded from http://www.realvnc.com.

YUV: YUV is the color space used to represent video pictures in three components. Y stands for the luminance component (the brightness) and U and V are the chrominance (color) components. YUV signals are created from an original RGB

(red, green and blue) source with a linear transformation. YUV representation separates color and brightness (black-white images). This lets easy manipulation of color images to reduce bandwidth.

Vitae

NAME: Ahmet Uyar

DATE OF BIRTH: January 1st, 1974

PLACE OF BIRTH: Karaisali, Adana, TURKEY

EDUCATION:

MAY 2005 Ph.D. in Computer & Information Science,
College of Engineering and Computer Science,
Syracuse University,
Syracuse, NY, U.S.A.

DECEMBER 1999 M.S. in Computer & Information Science
Department of Electrical Engineering and Computer Science, Syracuse University,
Syracuse, NY, U.S.A.

SEP 1996 B.S. in Electrical and Electronics Engineering,
Cukurova University,
Adana, TURKEY

EXPERIENCE:

Sept. 2001 – May 2005 Graduate Research Assistant
Community Grids Lab,
Indiana University
Bloomington, IN, U.S.A.

Sept. 2000 – August 2001 Graduate Research Assistant

	School of Computational Science & Information Technology, Florida State University Tallahassee, FL, U.S.A.
September 1999 – August 2000	Graduate Teaching Assistant Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, U.S.A.