A FRAMEWORK FOR CONFERENCE CONTROL FUNCTIONS

Nadia Kausar

A thesis submitted for the degree of
Doctor of Philosophy in Computer Science
University of London

June 2000

UCL

Department of Computer Science

Abstract

Conference control is an integral part in one-to-many communications that is used to manage and co-ordinate multiple users in different conferences. The degree which it controls the interaction of participants can vary in accordance with the conference structure in use. Conferences come in different shapes and formats but in recent years the two different views of conference control has been realised by two prominent standard bodies: International Telecommunication Union (ITU) and the Internet Engineering task Force (IETF). The former has focused on the centralised control of conferences (formal/tightly coupled). Whereas the IETF approach is referred as the informal/loosely coupled conference. However, in addition there are other conference controls for desktop multimedia system that are not yet standardised.

In this thesis the writer proposes a generic conference control system to provide a single mechanism that supports interoperability between two or more inherently different architectures mentioned above. The consequence of having very distinctive systems is that they are not interoperable. Each system provides a set of functionalities that are not easily provided in the other one. Users or administrators of such systems end up supporting dual stack technologies. However, both the systems have some common components like a call control and media part. In this research, one of the key tasks performed was to interwork two innate different architectures using their common call control functions. Also, a set of conference control functions have been derived and implemented that can be used by any conferencing systems over heterogeneous networks. The architecture that provides a set of functions and requirements that are visible or invisible to any user in any type of conferencing is termed as CCCS (Common Conference Control services) in this thesis.

In order to evaluate the integration of different services offered by different conference control model, IETF's Session Initiation Protocol (SIP) and ITU's H323 conferencing have been interoperated in an implementation. It has been established that conferees from completely different architectures can participate in one session by a mechanism such as CCCS which interoperates a set of services that are common in any conferencing systems. These common services have been derived by careful analysis, and with the aid of formal methods, state transition diagrams and a well-known software design pattern. The CCCS framework presented in this thesis provides the flexibility to users which are not present in just one of the architectures alone.

This thesis performs three key tasks: a) a detailed analysis of the activities appropriate for different types of sessions, and the common set of functions required to interwork them b) the transport level requirements to make these set of conference control functionalities reliable and scalable (in this case IP multicast have been studied) c) an effective solution to show extendibility afforded by the framework to charge conferences from the session layer. CCCS, designed with these functions have been tested to work over unreliable networks like the Internet and is used to draw some general guidelines for the design of such systems.

Acknowledgements

This thesis is the result of persistence, determination, and hard work. The four years that went in to produce this work received a lot of support from friends, family and UCL's members of staff. Without them I do not believe that I could have completed this work.

First of all, I would like to thank my supervisor Jon Crowcroft for the discussions and the idea exchanges that formed the backbone of this thesis. My UCL friends, Steve Wilbur, Graham Knight and the Networked multimedia team provided tremendous constructive criticism that earn my sincere gratitude. Ken Carlberg deserves very special thanks for believing in me and for being there when I needed help the most. Furthermore, I would like to thank Ian Marshall in BT labs for inspiring me with VoIP ideas and funding me for three years.

I would also like to thank my friends outside UCL (who managed to stay as friends after all these years of me ignoring them because of this PhD) and William Fahy for being supportive. My loving family always helped me with moral support and they always thought I could finish this PhD. Unfortunately my dearest father did not see my hardest effort come to an end. I dedicate this thesis to the loving memory of my father Prof. K. Ali (1920-1999) and my mother Prof. Azra Ali.

TABLE OF CONTENTS

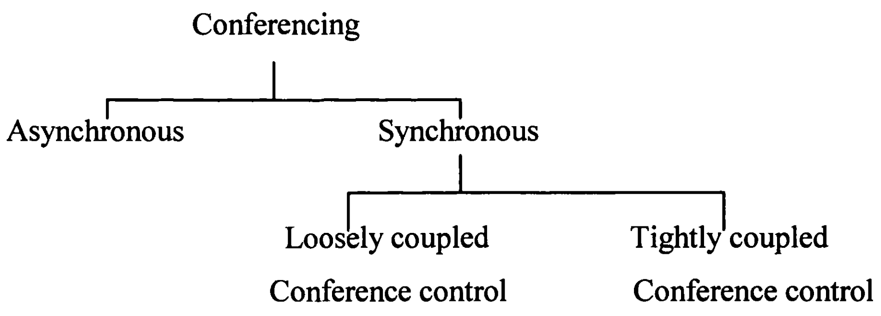
LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

This thesis presents a new framework for multimedia conferencing systems. The recent advances in computer technology and data networking have made video conferencing a popular medium for users to interact with each other from remote locations. The term "conferencing" is used in two different ways: firstly to refer to bulletin boards and mail list style asynchronous exchanges of messages between multiple users; secondly, to refer to synchronous or real-time conferencing including audio, video, shared whiteboards and other applications [Schooler93]. This thesis focuses on the architecture and the functionality required for the latter application and the former one will be ignored from now on.


Conferencing
Asynchronous
Synchronous
Loosely coupled Conference control
Tightly coupled Conference control

There are two main types of conferences: tightly coupled and loosely coupled. In the former, referred to as **Formal/tightly coupled** conferencing, the recipients are aware of who the other participants are and it is mainly used for business meetings, one-to-one sessions etc. That is to say, it employs a centralised control of activities. In contrast, in the **Informal/loosely coupled** conferencing the sender is not aware of who the receivers are, no strict floor control is necessary and it is used for broadcasting large research conferences for example and mainly followed by the research community [Kausar (a)]. Both models require a control mechanism which provides functions to establish, run, and terminate conferences as well as to allow the participants to move into and out of conferences, provide consistency etc [Kausar (b)]. In recent years, two different views of conference control have been proposed by two prominent standard bodies: the

International Telecommunication Union (ITU) and the Internet Engineering Task force (IETF).

The underlying architecture of these two types of conference is inherently different. The IETF model uses a distributed approach where no chairman role for a conference is necessary. The senders are not aware of all the receivers, so this type of conferencing can scale to a large number of participants. The ITU's model of tightly coupled conferencing is designed mainly for the circuit switched networks and different channels are used for the chairman role and the media distribution. The knowledge of all participants and the applications involved are necessary in this model. The consequence of having these two distinctive systems is that they are not easily interoperable. As a result users end up supporting several tools and multiple protocol stacks because the nature of a particular type of conference may change or the users may decide to use a different set of tools half way through a conference. For example, three participants in a research conference may decide that the conference they are currently holding should really be a brainstorming session and a lot more people are need to be involved. At present, it is very difficult to change from a tightly coupled conferencing scenario to loosely coupled and vice versa. Also it is difficult to manage and maintain various applications and operating systems associated with extremely different models and the difficulty of changing from one system to the other shows the inflexibility of today's conferencing model.

ITU and IETF have an extensive amount of work going on to produce different components of conferencing like different applications, tools and their features. Useful findings by [Handley M] and [Ott J] shows IETF based conferencing infrastructures where a structured conference control is absent and normally has been left to human consensus. In 1992, Schooler et al [Schooler92] focused on conference control as a way to provide intercommunication between separate tools. Conference control has not just been used as a way to co-ordinate users and different tools but as a desktop remote controller to set and change audio/video level, delay and volume etc [Perry] in remote locations. In the late 1990s, the ITU Study Group 16 produced a tightly coupled approach to conference control where different users have a common set of capabilities and minimum required resources to join a conference. But, there is little information available on the full architecture of different models of conference control and its management. A "common control" architecture will provide essential features and systems based on that will be easy to design, implement and maintain. The users then do not have to maintain multiple stack technologies in order to switch from one model to the other.

The purpose of this thesis is to examine how communication using different conference control mechanisms can be seamlessly integrated into a single mechanism, and to propose the Common Conference Control Services (CCCS) that provides a set of functions and services in a framework that is essential for any conferencing system. The aim to develop a communication infrastructure rather than a specific groupware application has several implications. First of all, no assumptions about the environment can be made [Ott J]. The infrastructure has to be portable across different hardware and operating systems platforms and has to provide the foundation necessary to achieve interoperability between different conferencing systems. Therefore, we propose a system that operates on the session layer. The other advantages of a session layer based protocols are : a) it is transparent to network support b) it has minimum impact on applications, and c) the design is extensible. As an example of extensibility, the design of the CCCS is able to support additional functionalities that are not directly applicable, such as charging [Kausar (c)].

There is no comprehensive set of design principles that would lead to a satisfactory solution of all the problems that can be observed for users in different tasks. However, as pointed out in [Sasse], designers of multimedia conferencing systems should not determine the nature of interaction between users, but should support many possible interaction styles and let the users decide which is most appropriate. At the same time, a set of defaults should be available for novice users. The architecture of CCCS uses this approach and the functions are implemented and tested over the Internet.

1.1 Group cooperation and different phases

Specific types of group cooperation can be decomposed into asynchronous collaboration, followed by a synchronous collaboration phase as shown in Figure 1.

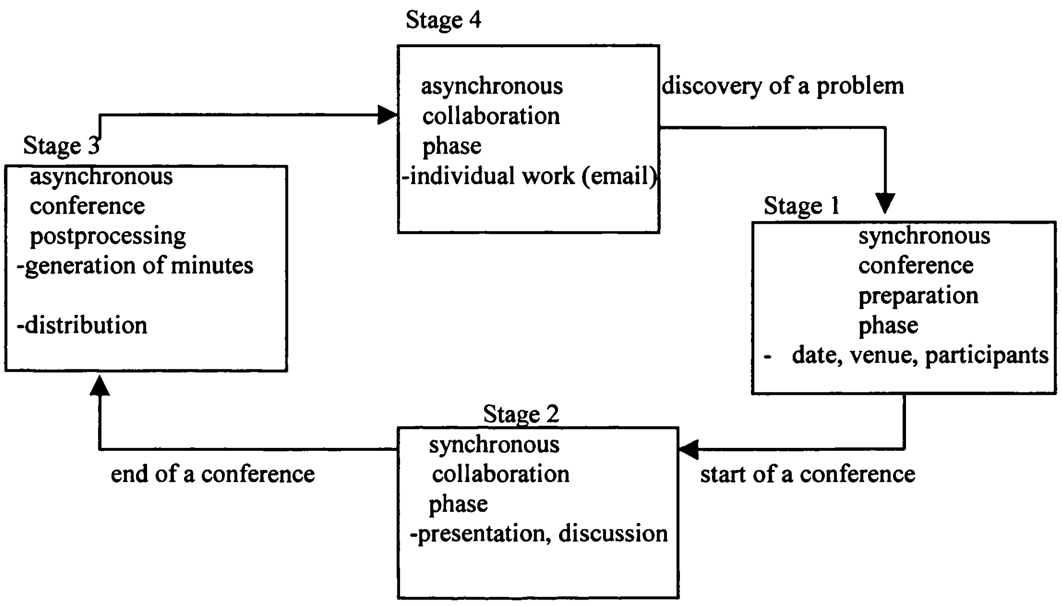

Stage 4
asynchronous collaboration phase
-individual work (email)
discovery of a problem
Stage 1
synchronous conference preparation phase
- date, venue, participants
start of a conference
Stage 2
synchronous collaboration phase
-presentation, discussion
end of a conference
Stage 3
asynchronous conference postprocessing
-generation of minutes
-distribution

Figure 1: The four phases of group collaboration

Synchronous interaction requires the presence of all cooperating users while asynchronous cooperation occurs over a longer time period and does not require the simultaneous interaction of all users [Rodden]. In this thesis we concentrate on the synchronous collaboration phase (stage 2). This phase covers the entire course of a conference with two or more participants. A conference may include presentations, discussions, reviews, voting, cooperating work sessions (e.g joint editing etc.). Due to the immediate interactions, the synchronous collaboration phase is also best-suited for updating group members about the latest developments, current project status, etc. Therefore, this cooperation phase is crucial to the entire process of group collaboration.

As Ott et al [Ott J] pointed out, in this idealized model, each activity of any of the group members may be assigned to one of the four phases, and the phases progress cyclically.

In reality, the borders between the phases are soft: individual phases may overlap with others. Depending on the situations, phases may also be very short or even left out entirely.

The next section gives an overview of conferencing and discusses different conference parameters and components and their positions in different layers of OSI.

1.2 Conference parameters and components

Figure 2 shows different meeting types that present the type of conference on the left hand side, a simplified correlation with meeting types, a set of meeting parameters [Schooler92] like interactivity between participants, and the number of participants that may be involved. These parameters and components make up a canonical model of conferencing which is the guideline for designing either a distributed or a centralised model of conferencing (discussed further in section 1.3).

In the following, a set of parameters that are shown in Figure 2 are discussed. These parameters may be used to describe the characteristics of all meeting types [Schooler92]:

- Interactivity - Traditionally, a highly interactive conference tends to be a telephone call, or a hallway meeting where number of participants is low and it has high coordination.
- Group size - Ranges from two to more than a million (TV broadcast), for most review type of meetings or small panel discussions, groups of up to twenty are common.
- Conference duration – Ranges from minutes to hours.
- Geographic distribution – participants could be located on the same floor, in the same building, conference room or different countries.
- Establishment – conferences can be established in an ad-hoc manner or in advance, it may start at a planned time or it may start when the invitee has joined the phone call. Big conferences like seminars are normally planned in advance whereas telephone calls take place in an ad hoc manner.
- Admission policy – Anybody may enter an open conference or a closed conference is limited to a predefined set of participants (may require passwords)

- Floor control -Floor control, a metaphor for "assigning the floor to a speaker", which is applicable to any kind of sharable resource within conferencing and collaboration environments. The tighter the floor control is, the conference type is classified more as a formal conference or tightly coupled conference.

Meeting type	Main characteristics	Interactivity	no of Participants[1]
Point-to-point call	high coordination	high	2
	rigid		
Hallway meeting	small sessions	high	2-6
Review meeting (e.g. boardroom meeting)	static	high	2-20
Classroom	less coordination	medium-low	10-100
Panel discussion		low	10-100s
Seminar			
	few policies	low	10s-1000s
Lecture	low overhead		
	Large sessions		
TV broadcast	dynamic (receivers change)		

Figure 2 Meeting Types [Source: Schooler [Schooler93]]

- Conference policy – A conference policy describes how strictly a conference is managed and issues of conductorship (as described below) and privacy issues. User characteristics influence a policy, for example, a policy might be user-dependant and role dependant. A chairperson role might allow a single person to mediate floor control decisions, or priorities associated with users might help resolve conflicting

[1] A common number is presented here, these numbers may vary in different situations

requests for control. A policy might implement a single global thread of control for a whole application, or it might provide independent control over individual parts of an application (i.e. objects) [Boyd]. A conference architecture has to accommodate both extremes as far as these policies can be actively supported by technology [Boyd]. Privacy policies control whether other clients can learn of the sessions existence, its attributes and the identities of the current session participants.

- Conduct management - A conference may be chaired by a participant (the chair person may be prearranged) or the conductor role may be passed around. A conference may also switch between conducted and non conducted mode depending on the task being performed.
- Coupling mode – Coupling mode refers to the consistency of state information about a teleconference at the various participating sites. In a tightly coupled scenario, all participants are aware of the participant list, their roles, what applications are there and why and how they are being used, e.g. a small meeting room. In a loosely coupled scenario, consistency is not explicitly provided for. Participant lists are not exactly known, all the applications may not be used by everyone at every site. An example of this could be a talk with an audience of hundreds of listeners.

The next section is a detailed overview of conference control and its association with different levels.

1.3 Conference Control Requirements and different perspectives

Most of the conferencing models need to fulfil some basic system requirements:

a) Application interaction – Applications for multimedia conferencing, e.g. video tool, whiteboard etc. need to be started with the correct initial state, and the knowledge of their existence must be propagated across all participating sites. Applications may need to cooperate (for example to achieve audio and video synchronisation)
b) Membership control – Who is currently in the conference ? [Handley]
c) Floor Management – Who or what has the control over the input to particular applications?

d) Network Management - Request to set up the connections between end-points and request from the network to change bandwidth usage.
e) Session management - Startup and Finish Conference, inviting people, joining, side sessions etc.
f) Ordering - Process ordering and ordering of the resource requests, so avoid all necessary deadlock situations.
g) Reliability – Sending applications need to be sure that the destinations received the messages. Whether all the messages sent out have to be acknowledged or the **n** out of **m** destinations picked up the message is a design issue within the conference control.

Depending on users' activities in different styles of conferencing, the services expected from conference control vary from application level to network level. So for example, in a tightly coupled conferencing membership control and its consistency are very critical to the user in the application layer. The network should be reliable to support this activity but it is not up to the network to provide that information. On the other hand, in a loosely coupled conferencing, the network must be able to handle a large number of participants. In this framework consistency arises slowly by virtue of the periodic updates sent by network nodes (details in Chapter 4). Table 1, 2 and 3 outline the different services provided by conference control at user, application and network level.

Table 1 outlines the services expected from two types of conference control and their differences from a user viewpoint. The users define conference policies and they expect the application (discussed in Table 2) to meet their requirements. As shown in Figure 2, as the number of participants increases, the activities are more likely to have less coordination and a looser floor control. In these cases the conference control tends to be left to human consensus. When there are only few people in a conference, users expect to have the exact list of participants and the applications that are in use. The conference control tends to be more formal in these scenarios.

Services	Formal/strict/tightly coupled	Informal/loosely coupled
Floor control	Requires a floor control, if one common video channel is there, it follows the floor of the conference so that it is the video signal originating from the current floor-holder site. Let a requester speak.	Human chair (if the policy specifies), but normally anyone can speak at any time.
Application interaction/control	Symmetric: require all participants to use the same set of media for communication	Asymmetric: receiver driven, whatever the recipients are capable of running
Membership control	Knowledge of all participants, at least the speakers in the panel sessions.	Senders do not have to know about receivers.
Session control	a) Need to have a strict start/finish time, date etc. b) The session does not start till the chairman initiates the conference.	Someone has to create a conference and start it, it does not matter when the receivers join.
Reliability	Guarantee of service required	Depends on the type of application/networks

Table 1: Human level of conference control

Table 2 shows different features of conference control in the application level. Conference control in this level is expected to provide user visible functions such as update a registry when a participant joins or leaves, request a floor via a user interface etc. It is also in the application layer where the conference control provides management functions such as interoperability between non-compliant software, address translation and refuse a floor to an unauthorised speaker.

Services	Formal/strict/tightly-coupled	Informal/Loosely
Floor control	Checks the Registry. If the requester is a valid speaker assign it to him.	May not be needed
Application Interaction/control	Exchange capabilities to start with. Check application registry, If valid control application session.	Capabilities exchange still applies, but no application control.
Membership control	Update the registry. Includes Join, Leave, Invite, Exclude and authentication control.	Use RTCP [RTP] to inform about the participants
Session control	Profile definition(conference name, description, password protected/not, listed/unlisted, conductible/nonconductible), setup & termination, tear down connection at finishing time	Conference name, password (if applies), description, media to be used, setup & termination by the initiator
Reliability	The floor control and session control needs to be run on TCP/ some reliable transport protocol	Depends on the policy of the conference

Table 2: Application level of conference control

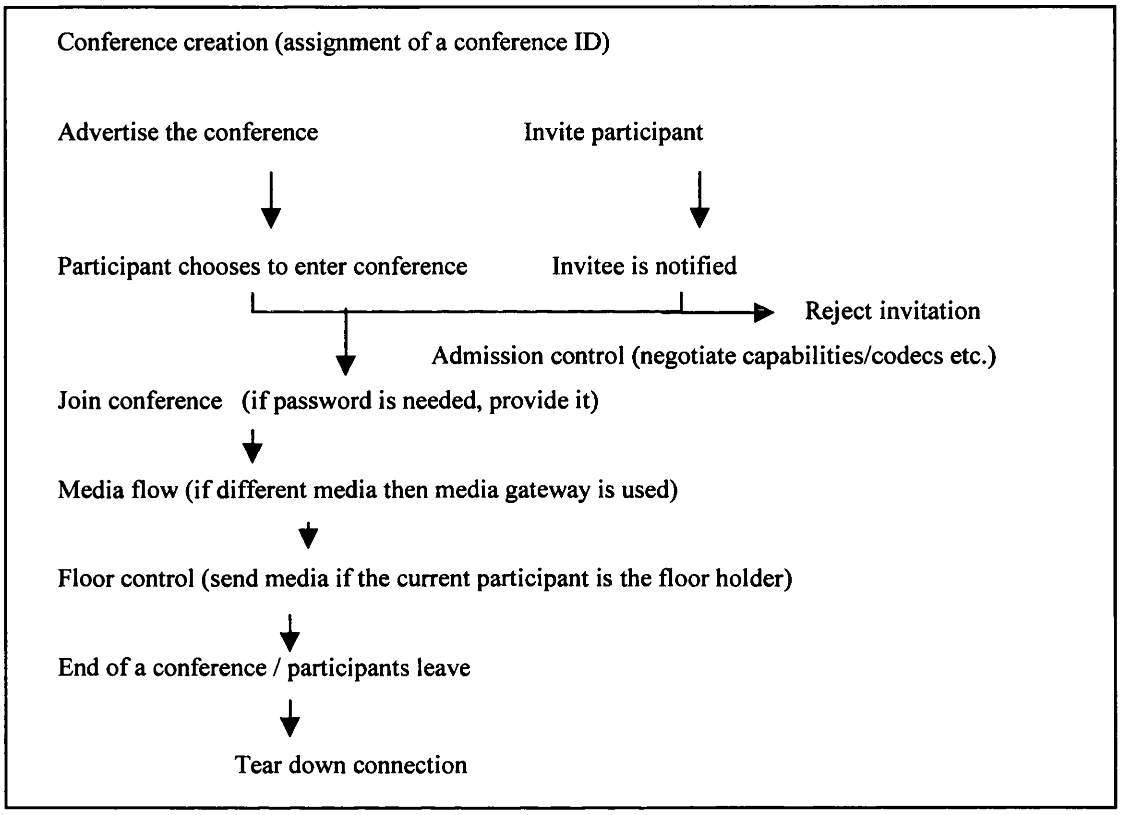
Different types of conferences are designed for different types of networks. There are point-to-point, point-to-multipoint and multipoint-to-multipoint conferences which may or may not run on packet based networks (e.g. Internet) or circuit-switched networks (e.g. ISDN/PSTN). It is not possible to design the services required from all types of networks. We have taken into consideration which networks may provide a particular type of conference and the methods for it:

Criteria	Pt-to-pt	Pt-to-multipt	Multipt-to-multipt
Network type	Circuit switched networks e.g. ISDN / PSTN	Circuit switched networks e.g. ISDN/PSTN Internet Protocol (IP)	Internet Protocol (IP)
Methods of building connection	Explicit channels Unicast	Explicit channels Unicast/Multicast	Multicast/ Broadcast
Media transmission	Usage of Multiplexers, Reliable channels and ITU standardised frames	Same Real Time Transport Protocol (RTP/RTCP)	IP based datagram, Real Time Transport Protocol (RTP/RTCP)
Reliability	Explicit signalling TCP	Explicit signalling TCP or Reliable Multicast Transport Protocol e.g. SRM	Reliable Multicast Protocols e.g. SRM

Table 3: Network level of conference control

In order to conduct a point-to-point or point-to-multipoint conference, we need to create certain types of channels which convey the services. The conferences based on Circuit Switched networks have separate channels for call connection/control and media transmission. For example, for two people to exchange audio, two logical channels must be opened, one from caller to callee, and vice versa. Also in order to carry media, two different channels must be opened from both sides. This is done to ensure reliable delivery of media.

To summarise, the tightly coupled conferencing normally seen in ITU emphasizes reliability, tighter policies and centralised control whereas the latter type of conference seen in IETF is less concerned about tight co-ordination and reliability, instead it focuses on scalability and it employs fewer policies. In Chapter 2, the centralised and the distributed models of conferencing and their different issues mentioned above are further discussed. However, this leads to the question of what a canonical model of conferencing looks like. A typical desktop conferencing always shows certain activities as seen in Figure 3.



Conference creation (assignment of a conference ID)
Advertise the conference
Invite participant
Participant chooses to enter conference
Invitee is notified
Reject invitation
Admission control (negotiate capabilities/codecs etc.)
Join conference (if password is needed, provide it)
Media flow (if different media then media gateway is used)
Floor control (send media if the current participant is the floor holder)
End of a conference / participants leave
Tear down connection

Figure 3: Activities in a canonical conference

A conference needs to be configured by either a system or a user with the following parameters: subject topic, estimated number of participants (or a precise list of participants), policies such as admission policy, charging policy (if applicable), security and floor control. Then depending on the policy either the conference is advertised or the participants are invited. When participants start to join the conference, admission and security policies are applied and negotiation of capabilities take place. Therefore, the participants might be required to pay a certain fee, or provide a password to join and need a common set of tool to participate. Once the media such as audio and video start to flow there may need to be mechanisms to provide desired quality of service and a media gateway to provide a consistent format of audio and video. Floor control will ensure that the requestor gets to speak or transmit data and this process is fair (i.e. consistent with the conference policy). The minimum quality that is expected from a conference is the reliable delivery of data. After the conference terminates, the appropriate channels for media or signalling are successfully closed down.

Observation shows a canonical model of conferencing tends to follow a pattern which is described above, which proposes that the minimum number of essential activities that a control mechanism needs to cater for are:

- Creation of a conference
- Facilities to join a conference
- Invite a participant
- Floor control
- Reliability
- Leave a conference

Most of the activities shown in Figure 3 are supported in CCCS proposed in this thesis.

1.4 Thesis statement and objectives

The main objective of this thesis is to present a generic framework for multimedia conferencing. To identify and observe different coordination and control tasks associated with separate conferencing model and propose an ideal way to integrate and support these tasks. Identify parts in specific architecture where components are missing and where components that can be unified easily.

This research work sets out to propose a framework, Common Conference Control Services (CCCS) to show that it is possible to develop a generic framework for conference control and conferencing. During the research a number of sub-objectives became clear which makes this framework more effective:

- Transport protocol
- Reliability
- Allocation of resources for a successful conference
- Billing

CCCS framework has been designed to be extensible. In the future when there will be a requirement to add billing and authentication services as an integral part of commercial conferencing, CCCS is flexible enough to cater for that. The architectures we have today for conferencing do not support additional services to be added in the easiest manner. In this thesis a new charging mechanism is proposed as a value added service for conference control which can be integrated in the proposed framework.

The framework proposed needs to support major existing conferencing architectures from networks perspective to be effective. A network needs to assist the CCCS to provide its main services reliably in a scalable manner. Reliable data delivery for conferencing control functions in a heterogeneous environment like the Internet model has not been investigated and lacks a solution. A suitable transport protocol has been proposed that will provide the requirements of conference control, like congestion control, membership control and reliability over the Internet.

Finally, an implementation needs to be carried out to validate the concept of interoperability of different conferencing models.

1.5 Thesis Outline

In Chapter 2, related work is discussed, mainly giving an overview of IETF and ITU models of conferencing; its design, architecture and distinctive models of conference control associated with these architectures are also reviewed. A summary of both models of conferencing is provided at the end of the chapter. We did not consider it appropriate to provide an extensive overview of much earlier work that assisted in the evolution of the conferencing frameworks we have today, since this is covered elsewhere [Boyd]. However, we will mention early work where we consider it relevant.

Chapter 2 sets the scene for Chapter 3. From the summary in Chapter 2 it will be shown that the conference architecture we have today is not flexible enough to allow different models of conferencing in heterogeneous environments to be interworked. Therefore, a generic framework for conference control referred as CCCS, that can provide that functionality will be introduced in Chapter 3. In this chapter, two main features of conference control protocol are identified and a framework in that reference is developed. A framework that can interoperate between two innately different models of conferencing has been implemented as a proof of concept. Implementation details such as message format, transport services, error control and performance is also discussed here.

In Chapter 4, reliable network protocols, especially reliable IP multicast protocols are reviewed. A generic framework for conference control such as CCCS needs to provide its services reliably. Therefore, a reliable network is required to support this framework to be effective. The conferencing architectures seen in ITU today provide reliability but not scalability, whereas in the IETF model of conferencing we experience the contrary. Therefore, a network protocol that can support a large number of participants but be reliable at the same time is vital. Reliable IP multicast introduced in 1990 [Deering] shows a promising solution for future conferencing and therefore in this chapter we introduce some of the available protocols.

In Chapter 5, some of the reliable multicast transport protocols described in Chapter 4 are analysed for the purpose of conference control. In other words, most suitable features and functionality provided by reliable IP multicast protocols for some crucial features of conference control are identified. We do not propose a new network protocol but propose some adaptations to the most suitable existing network protocol.

The design of CCCS allows to support additional functionality which are not an integral part conferencing today, such as billing and charging. As an example of its extensibility, a value added conference control service charging is proposed in Chapter 6. Facility to charge users for Internet conferencing in a commercial environment adds an attractive feature for providers. It covers the cost for providing services and it also controls the behaviour of users. In this chapter, difference charging mechanisms on the Internet are reviewed and then a session based charging mechanism is proposed which fits in the CCCS framework.

Chapter 7 offers our conclusion, and identifies some issues outside the scope of this thesis that would benefit this work in the future. Finally a summary of main contributions are presented.

Chapter 2

Related Work

This thesis and the related work that we discuss here, concentrates on different models of conferencing, its structure and requirements. In this chapter in particular we discuss design, implementation and requirements of conferencing in Application and Session layer. We do not discuss network or transport layer requirements in details here.

Computer based, desktop conferencing systems help people to interact with each other from their homes or offices by exchanging information in several media. These systems support real-time conversation through the co-ordinated exchange of voice, video, and computer program output (e.g., text, graphics, and spreadsheets) [Ahuja]. Conference participants can share multimedia information, simultaneously view and update information if required. The participants can be close to one another, exchanging information over a local area network (LAN), or geographically separated, transmitting information over wide area networks (WANs). A common goal of these conferencing systems is to emulate important characteristics of face-to-face conversations. The catch with personalizable conferencing is that it is difficult, if not impossible, for designers to predict ahead of time the complete range of behaviours the system should exhibit. One solution is to make the system flexible, so that new behaviours can be created and old ones modified in ways that the original designer had not anticipated.

The humans participating in a conference generally need to have a specific idea of the context in which the conference is happening, which can be formalised as conference policy (as mentioned in Chapter 1). Some conferences are essentially crowds gathered around an attraction, while others have very formal guidelines on who may take part (or listen in) and who may speak at which point. In any case, initially the participants must find each other, i.e. establish communication relationships (conference setup). During the conference, some conference control information is exchanged to implement a conference policy or at least to inform the crowd of who is present. In addition, security measures and interoperability may be required to enforce the conference policy.

In this chapter, we review how conferencing architectures have developed and evolved to meet the above mentioned criteria. We primarily focus on the circuit switched ITU and the packet switched IETF model of conferencing, since these two models of conferencing dominate the research community and the commercial industry. In the process, the shortcomings and the strengths of both models are pointed out. Then we focus on other conferencing architectures and conference control protocols that have been researched and implemented. Finally we look at some gateway protocols that are currently being analysed and implemented as a result of a dramatic change in telecommunication industry. The most significant change in the last twenty years has been the move from using circuit switched networks for multimedia data towards packet-switched networks and the Internet in particular for multimedia data [Handley M]. More recently, ITU's original design of conferencing has evolved to include packet networks, and a more loosely coupled and distributed approach to conference control.

2.1 The ITU architecture for multimedia conferencing

The design of ITU's conferencing and the controls associated with it originated in person-person video telephony, across the POTS (Plain Old Telephone System) or its digital successor, ISDN. This is a circuit model, where one places a call using a signalling protocol with several stages and the link resources are allocated at the call setup. The resources such as communication channels are released at the end of a call. The general assumption is that all participants are consistent about the status of other participants, applications and applications' capabilities.

The ITU multimedia conferencing architecture essentially consists of two parts: (low level) network specific protocols for establishing and multiplexing physical connections; and a (high level) network independent conferencing infrastructure for multipoint communication, conference control, and support of conferencing applications [Ott J].

The low level protocols have been primarily designed for point-to-point audiovisual communication, in particular video telephony. They define the communication procedure for placing a multimedia call between two systems which are network dependant. The low level protocols which are network specific are developed by Study Group 15 of the ITU[2] and are defined in several series of recommendations:

- The H.310 series of recommendation for ATM networks,
- The H.320 series of recommendations for ISDN
- The H.321 recommendation for use of H.320 on top of ATM
- The H.323 series of recommendations for corporate (inter) networks, and
- The H.324 series of recommendation for analogue telephone networks and mobile systems.

Along with the definitions of the system components, system operation, and communication procedure, a set of audio and video encoding recommendations has been defined[3]. Wenger [Wenger95] discusses various encoding schemes and especially the H.320 series of recommendations.

2.1.1 T.120 recommendations

The high level conferencing infrastructure for ITU is described in the T.120 series of recommendations. Its development started around 1990 with the first recommendation of the series being approved by the ITU in 1993. At the time of writing, ten recommendations have been approved that define multipoint communication and conference control functionality as well as application protocols. The T.120 model is composed of a communications infrastructure and application protocols that make use of it. The ITU Secretariat maintains a list of the currently valid ITU Recommendations

[2]The work started in the late 1980s for video telephony over ISDN

[3] Audio encodings include G.711, G.722, G.723.1, G.728 and G.729; video encodings comprise H.261, H.262 and H.263

(T.121 - T.127). Each layer provides services to the layer above and communicates to its peer(s) by sending Protocol Data Units (PDU) via services provided by the layer below.

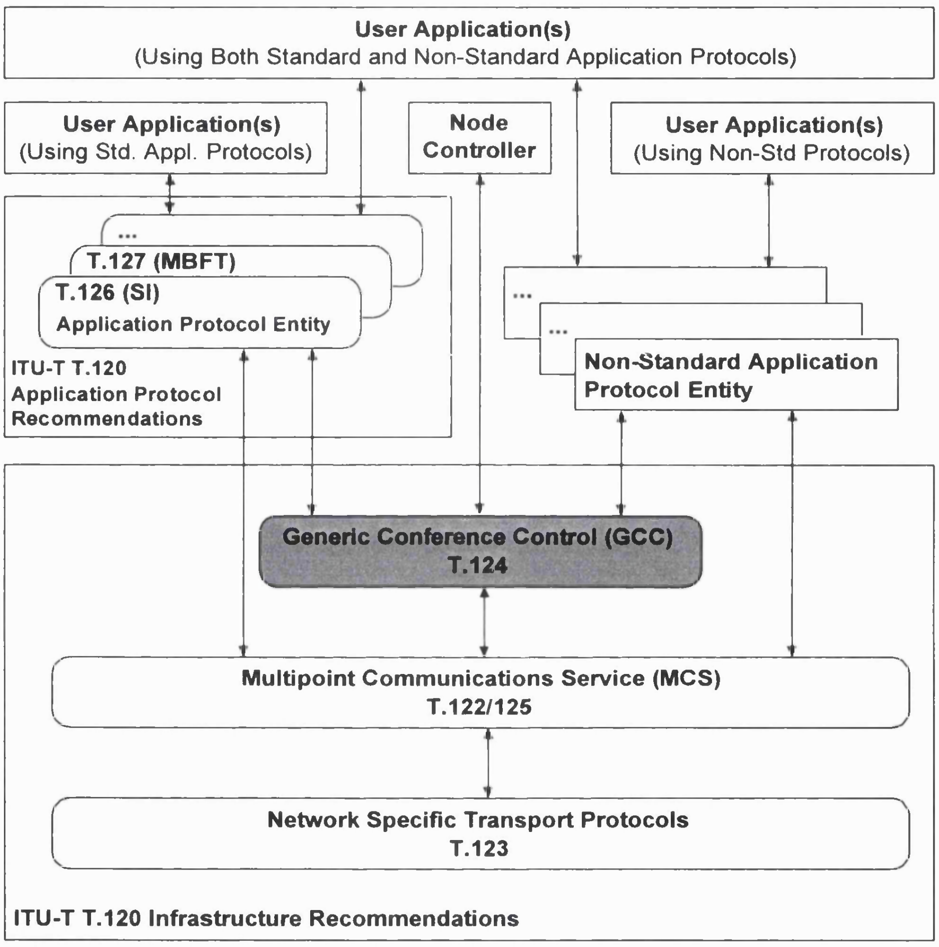

User Application(s)
(Using Both Standard and Non-Standard Application Protocols)
User Application(s)
(Using Std. Appl. Protocols)
Node Controller
User Application(s)
(Using Non-Std Protocols)
...
T.127 (MBFT)
T.126 (SI)
Application Protocol Entity
ITU-T T.120
Application Protocol Recommendations
...
...
Non-Standard Application Protocol Entity
Generic Conference Control (GCC)
T.124
Multipoint Communications Service (MCS)
T.122/125
Network Specific Transport Protocols
T.123
ITU-T T.120 Infrastructure Recommendations

Figure 4: T.120 recommendation infrastructure

2.1.1.1 Different components of T.120 based conferencing

In a T.120 conference, nodes connect up-ward to a Multipoint Control Unit (MCU). The MCU model in T.120 provides a reliable approach that works in both public and private networks. Multiple MCUs may be easily chained together in a single domain. Figure 5 illustrates a potential topology structure.

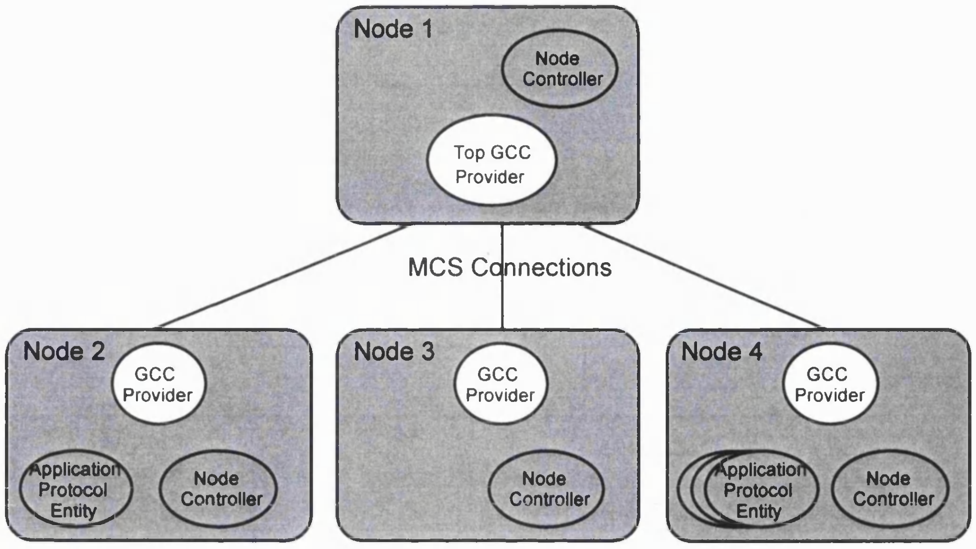


Figure 5: Hierarchy structure of T.120 nodes connecting up to a single MCU

As it can be seen from the above diagram, each domain has a single Top Provider or MCU (marked as Node 1) that houses the state information of all other nodes in the domain and is critical to the conference. If the Top Provider either fails or leaves a conference, the conference is terminated. If a lower level MCU (i.e., not the Top Provider) fails, only the nodes on the tree below that MCU are dropped from the conference.

- **Generic Conference Control (GCC)**

T.124 Generic Conference Control provides a comprehensive set of facilities for establishing and managing the multipoint conference. It is with GCC that we first see features that are specific to the electronic conference. At the heart of GCC is an important information base about the state of the various conferences it is servicing. One node, which may be the MCU itself, serves as the Top Provider for GCC information. Any actions or requests from lower GCC nodes ultimately filter up to this Top Provider.

Using mechanisms in GCC, applications create conferences, join conferences, and invite others to conferences. As endpoints join and leave conferences, the information base in

GCC is updated and can be used to automatically notify all endpoints when these actions occur. GCC also knows who is the Top Provider for the conference. However, GCC does not contain detailed topology information about the means by which nodes from lower branches are connected to the conference.

Every application in a conference must register its own application ID with GCC. This enables any subsequent joining nodes to find compatible applications. Furthermore, GCC provides robust facilities for applications to exchange capabilities and arbitrate feature sets. In this way, applications from different vendors can readily establish whether or not they can interoperate and at what feature level. GCC also provides conference security. This allows applications to incorporate password protection or "lock" facilities to prevent uninvited users from joining a conference.

Another key function of GCC is its ability to prevent conflicts for resources such as tokens and channels. Its ability to manage shared resources ensures that applications do not step on each other by attaching to the same channel or requesting a token (such as floor holder token) already in use by another application. Finally, GCC provides capabilities for supporting the concept of strict floor control in a conference. GCC allows the application to identify the floor controller and a means in which to transfer the holder's token.

The design and the structure of T.120 based conferencing imply that when the number of clients and applications increase, the size of GCC's register containing information of capabilities and various features of clients grow with it. The Top Provider becomes a bottleneck and the performance of the MCU decreases. In order to address some of these problems, T.124 has been revised.

- **T.124 Revised**

The ITU has completed a draft revision of T.124. The new version, called T.124 Revised, introduces a number of changes to improve scalability. The most significant changes address the need not to distribute GCC's registry information (referred to as its roster) to all nodes participating in a conference any time a node joins or leaves a conference. This registry contains detailed information on applications, other nodes' capabilities and their details.

To improve the distribution of roster information, the concept of Node Categories was introduced. These categories provide a way for a T.124 node to join or leave a conference without affecting the roster information that was distributed throughout a conference. In addition, the Full Roster Refresh, which was previously sent any time a new node joined a conference, was eliminated by sending out roster details from the Top Provider. These changes will not affect backward compatibility to earlier revisions of T.124.

2.1.1.2 Sequence of actions in T.120 based conferencing

The pattern of a T.120 based conference is:

a) Create the conference
b) Invite a participant to the conference
c) The participant accepts and joins the conference
d) Update the roster/application/membership registry
e) Terminate the conference

Each node in a T.120 based conference consists of three main functionality: a) node controller b) GCC provider and c) Multipoint Communication Services (MCS) provider, which allows the node to be capable of participating in multi-point conference. In a conference, multiple nodes (or MCS nodes) are logically connected together to form what T.120 refers to as a *domain*. When an MCS node wants to join a domain it has to send its request to the Top GCC provider (i.e. MCU) of the domain, which has to check the MCS capability of the requesting node to ensure that this particular node can join this specific domain. If the request is successful, the MCS provider of the MCU sends a confirmation to the requesting node. The number of messages generated just to be able to join a conference can be up to 16 [T.120]. Since, this type of conference was initially designed to work on a circuit switched network on a point-to-point basis, the specification tends to be very complex.

2.1.2 H.323 conferencing

H.323 [H.323] is an umbrella recommendation from the ITU which is designed to extend traditionally circuit based audio visual and multimedia conferencing services into packet (i.e. IP-based) networks. The H.323 system aggregates a number of standards, which together allow establishing and controlling point-to-point calls as well as multipoint conferencing. It provides a tightly controlled communication model, with explicit control and media connections set up between participants.

Although H.323 is minimally defined to operate utilising only peer H.323 terminals, the recommendation defines a number of additional logical H.323 elements. These elements are: terminals, gatekeepers, gateways, and multipoint control units (MCUs). Their main functions are:

- Terminal- clients/endpoints that provide two-way communication
- Gateway- an optional element to provide interoperability (H.320 compliant terminals over (ISDN, H.324 over PSTN)
- Gatekeeper – performs a number of tasks: address translation, bandwidth management, and charging for calls.
    - Address translation – gatekeeper acts as a central point for a H.323 zone address translation from LAN aliases to IP or IPX.
    - Bandwidth management – it is up to the gatekeeper to ensure that calls made to H.323 terminals via a gatekeeper have enough bandwidth to complete the call.
    - Charging – work is currently in progress [Gary] to allow calls to be routed to the gatekeeper so that they can be charged.
- MCU- supports conferences between three or more terminals, mainly consists of MC (multipoint controller) and MP (multipoint processor). MC handles capability negotiations and MP deals with media streams like media mixing, switching and processes conversion between different codecs and bit rates

Although H.323 clearly defines services and interactions between all of these logical elements, there are no specific hardware or software requirements mandated. Figure 6 depicts an environment of H.323 in terms of the logical system components and also shows a sample network topology which interoperates with the circuit switched networks (ISDN, PSTN) based conferencing

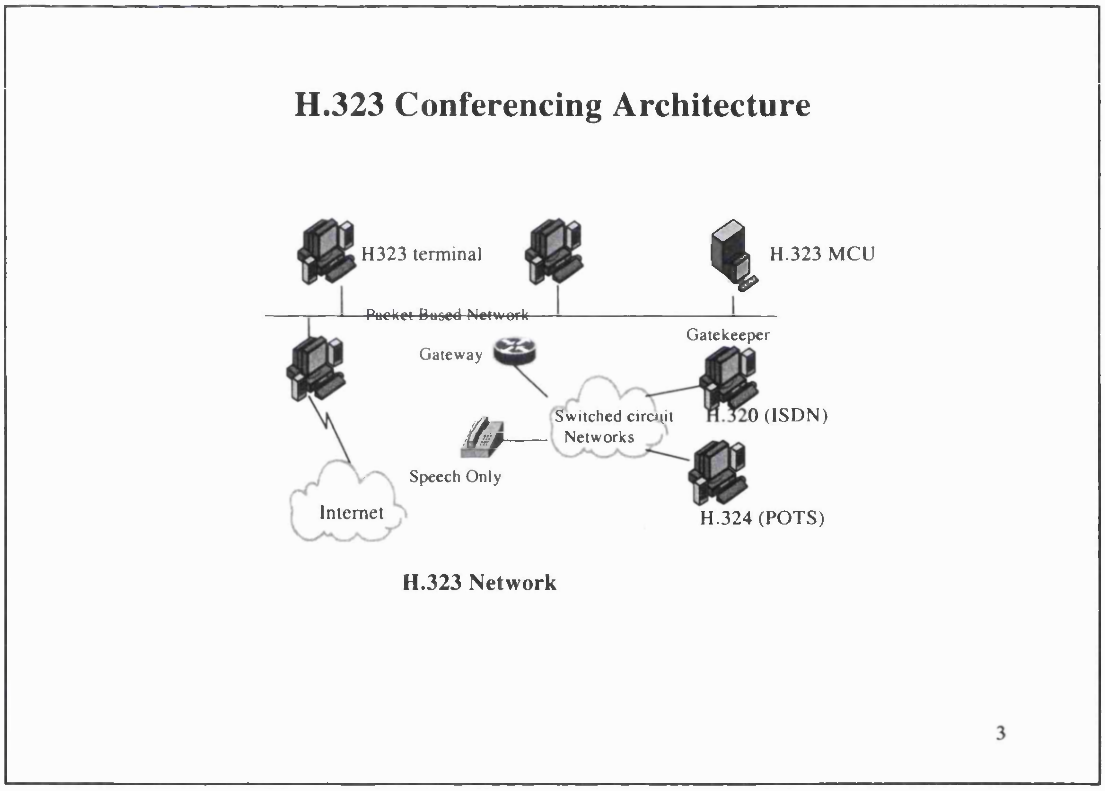


Figure 6: Environment of H.323 and sample network topology

2.1.2.1 Scope of H.323

Figure 7 illustrates a conference stack of H.323 showing all the core protocols. Although H.323 is specified to work over any packet based networks, only IP network is currently best in practice. In order to ensure reliable delivery of call setup signalling over IP, TCP is used while for transporting audio and video, unreliable connection (UDP) is specified. T.120 specification is used for data.

The initial connection is made from the caller to a well-known port on the callee. H.323 has specified the call signaling and bearer capability control in two separate protocols, H.225 and H.245. H.225 [H.225] covers the call setup and the initial call signalling. The H.225 document embodies two sub protocols: Registration, Admission and Status (RAS) and the call control messages are derived from Q.931. H.245 [H.245] is the media control

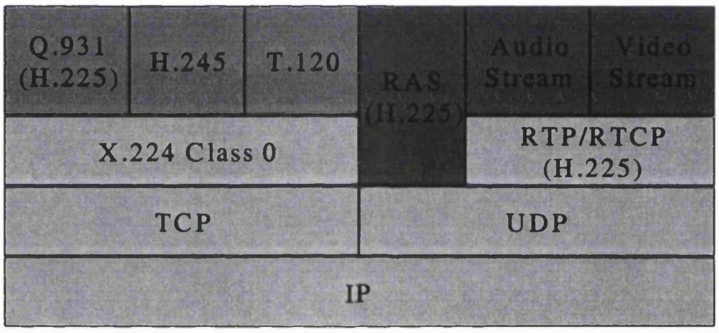


Figure 7: Conference stack for H.323

protocol that H.323 system utilises after the call establishment has completed. The addressing information required to create the separate H.245 protocol channel is passed in the call control message during the Q.931 call establishment phase. H.245 is used to negotiate and establish all of the media channels carried by RTP/RTCP. The H.245 protocol forms the common basis for media and conference control for a number of ITU multimedia communication systems. The functionality offered by H.245 that is used by H.323 falls into four categories:

a) Master slave determination
b) Capability exchange
c) Media channel control
d) Conference control

2.1.2.2 Sequence of operations in H.323 based conferencing

When the H.323 based conference runs over UDP it uses RAS (Registration, Admission and status) to setup the call. When the underlying transport protocol is TCP, the call setup messages are sent on the first TCP connection and the caller establishes the callee. Four setup messages: Setup, Alerting, Connect and Release Complete are necessary. Their operations and syntax are defined in detail in the Q.931 and H.225 specifications. The call signalling specified in H.225 follows a set of setup messages through a reliable channel which is based on the Q.931 protocol. H.245 uses the reliable H.225 call signalling channel to perform H.245 control functions, such as master/slave determination and conference control. During the terminal capability exchange procedure, both ends are based on their own terminal capability to set up the mutual acceptable bearer service to the network. Then logical channel(s) are opened to provide the bearers, for example, two channels for video and one for audio. This procedure is very similar to the bearer capability request in the ISDN Q.931 and SS7 ISUP.

Figure 8 shows the sequence of actions for setting up a conference in H.323. The caller sends *Setup* to initiate the conference immediately after establishing TCP connection. The S*etup* message contains the caller's name and IP address. The callee sends *Alerting* after notifying the user of the incoming call, if the call will not be accepted without user intervention. It uses the *Release Complete* message if it refuses the call. Otherwise, the connect message contains the address and port on which the callee is listening for the H.245 connection.

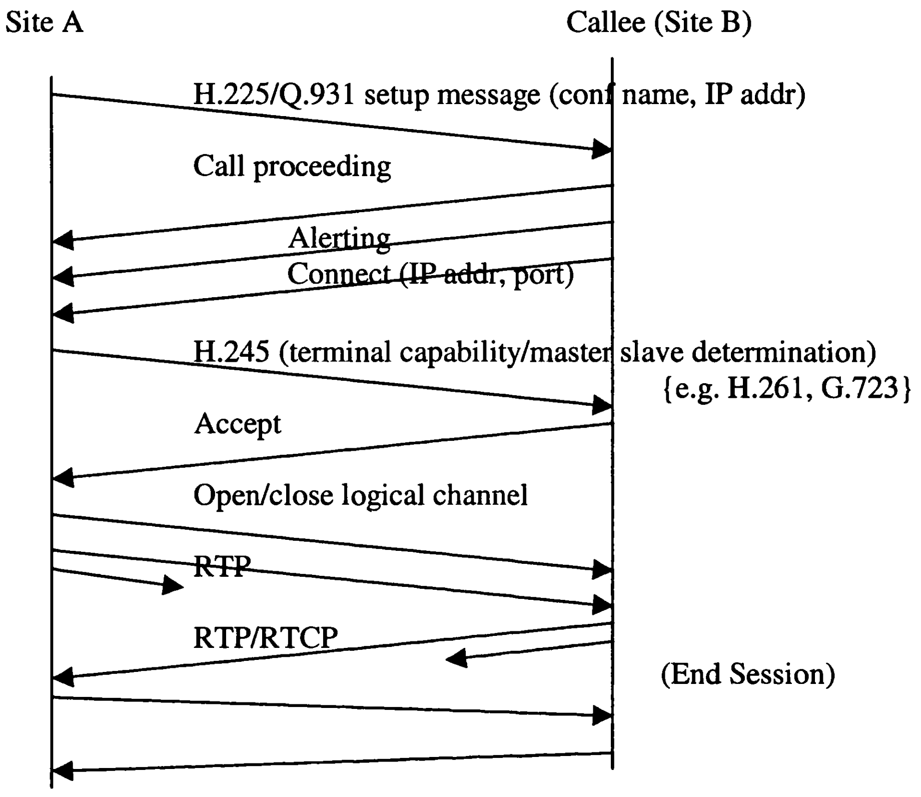

Site A
Callee (Site B)
H.225/Q.931 setup message (conf name, IP addr)
Call proceeding
Alerting
Connect (IP addr, port)
H.245 (terminal capability/master slave determination)
{e.g. H.261, G.723}
Accept
Open/close logical channel
RTP
RTP/RTCP
(End Session)

Figure 8: Message sequence in a H.323 based conferencing

2.1.2.3 Netmeeting – A product based on H.323

NetMeeting from Microsoft supports the H.323 standard for audio and video conferencing, which includes the H.263 video codec. NetMeeting enables users to share applications, exchange information between shared applications through a shared clipboard, transfer files, collaborate on a shared whiteboard, and communicate with a text-based chat feature. Also, its support for the T.120 data conferencing standard allows NetMeeting to interoperate with other T.120-based products and services. The Microsoft Internet Locator Server (ILS), uses an LDAP (Lightweight Directory Access Protocol) interface for NetMeeting directory services. Users can view the ILS directory from within NetMeeting or from a Web page, and review a list of people currently running NetMeeting. Then, they can choose to connect to one or more of the listed people or select another person by typing their location information. One can access the ILS and perform server transactions such as logging on and off and creating a directory listing of available users.

2.1.3 H.Loosely Based Conferencing

ITU based conferencing does not scale to very large numbers of participants because it uses a centralised architecture relying on a Multipoint Control Unit MCU to distribute media or data to multiple clients. The concept of H.Loosely coupled conferencing has been introduced to address the scalability problem. The main idea is to separate passive "listening" participants from interactive participants.

As shown in Figure 9, there is a panel in this type of conferencing, which consist of active participants who are senders and receivers of data and/or audio. The main principles of tightly coupled conferencing apply to the panel members. This panel consists of a small H.323 conference connected to a large number of RTP receiving terminals via RTP/RTCP. Within the panel, full interaction is allowed. Interaction could be through social- or chair- control.

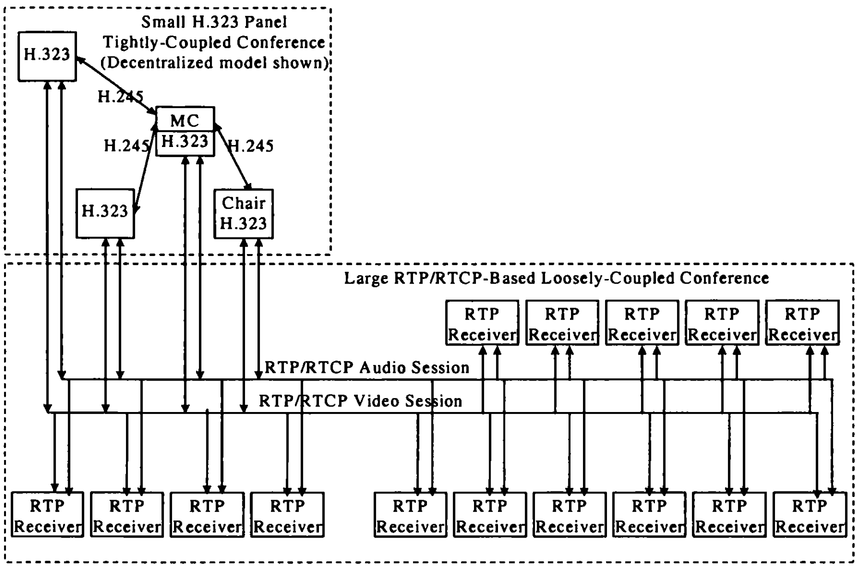


Figure 9: A large conference consisting of an H.323 Panel and RTP/RTCP based Receivers

Outside the panel, the participants are passive; they are essentially receivers who are not allowed to interact. If they wish to interact, they have to join the panel or get invited by the panel. Inside the panel any H.323 model - centralised, decentralised, or hybrid - can be used. But outside the panel multicast is used in order to provide the scalability required for the H.Loosely coupled conference. This can be achieved by using a MultiPoint MP to multicast media streams to the RTP receiving terminals.

- **Summary**

The infrastructure for ITU standardised conferencing follows a very formal policy where call setup emulates the design used in the POTS networks. The functionality assessed from this type of conferencing can be listed as follows:

- A strict conference control functionality to enforce policies;
- Reservation services for conferences and conference resources
- Support of conductorship and (for conducted conferences) of floor control
- Administrative functions for managing applications within the conference including capability negotiation, resource arbitration and address translation (from telephone network to IP netowork)
- Application protocols (e.g. T.127) for whiteboard, file transfer and remote device control
- A high layer entity (such as MCU) is responsible for organising nodes in a hierarchy based conference

2.2 The IETF architecture for multimedia conferencing

Although H.Loosely was trying to solve the scalability problem of the ITU based Internet conferencing, it still uses a central point like MP to distribute data to multiple clients. Also, if a conference consist of a large number of active participants, a tightly coupled panel cannot be formed and the H.Loosely coupled model is no longer effective. The IETF conferencing architecture eliminates this scalability problem by being completely distributed.

Various working groups have contributed to the development of a teleconferencing architecture in the IETF. These efforts started in the late 1980s with the work on multicasting for the Internet protocol in order to support group communication [Deering91]. Afterwards, improvements to the multicast routing infrastructure – the Multicast Backbone, Mbone (see section 2.3.1) [Mbone] have been achieved and resource reservation mechanisms to achieve a reasonable service quality have been worked out [Zhang93]. The Mbone has been used for a number of applications including multimedia (audio, video and shared workspace) conferencing. These applications involved vat (LBL's Visual Audio Tool), ivs (INRIA Video Conferencing system), rat (UCL's Robust Audio Tool), nv (Xerox's Network Video Tool), wb (LBL's shared whiteboard), and vic (LBL's) among others. The main distinctions between this type of architecture and the ITU's video conferencing architecture are a) in the IETF based conferencing architecture, all the protocols use a distributed architecture b) these protocols mainly work over IP and the control of conferences is very loosely coupled.

2.2.1 IETF's basic requirements and design

As mentioned above, the IETF based conferencing architecture consists of several independent protocols which operate in a stand-alone manner. For example, the protocol that is used to advertise a conference is called SAP (Session Announcement Protocol), and RTP (Real-time Transport Protocol) is used to carry audio and video. The design of the IETF's conferencing system have been made in a modular fashion. In ITU's architecture, an H.323 conference combines the audio, video, invitation functions (H.225) and conference control (H.245) to provide an integrated solution for a conferencing. In contrast, IETF's audio tool is separated from the video tool and the protocol for advertising can be separated from the signaling protocol (Session Initiation Protocol).

The protocols used in the IETF conferencing architecture are designed such that conferencing will scale effectively to large numbers of participants. It is clear that, with the unreliable network such as the Internet, these applications cannot achieve complete consistency between all participants, and so they do not attempt to do so [Handley]. The conference control they support usually consists of:

- Periodic (unreliable) multicast reports of receivers
- The ability to locally mute a sender if one does not wish to hear them.

2.3 Different protocols

The following sections describe all the major components and the protocols that contribute towards the IETF based conferencing architecture.

2.3.1 Mbone

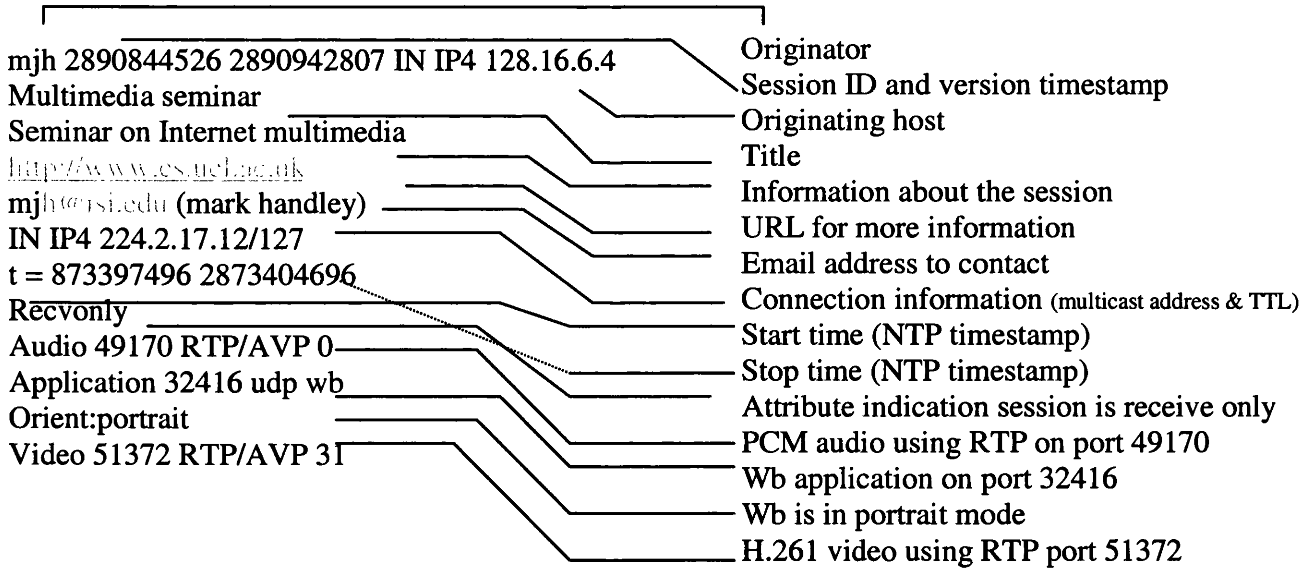
The multicast backbone, or Mbone [Mbone], implements IP multicast. During 1991, IP multicast was deployed on the DARTnet testbed network. In spring 1992, the DARTnet's native multicast network was extended to cover a small number of additional sites by tunnelling IP multicast across parts of the Internet without native multicast support. As a temporary measure, the Mbone (Multicast Backbone) was put together to allow reception of live audio from the IETF meeting in San Diego. Forty subnetworks in 4 countries and three continents were able to receive audio and talk back to the meeting. Although the audio quality was poor, the principle had demonstrated, and sufficient interest shown that the Mbone was not taken down again.

When a host wants to join an Mbone session or a multicast group in general, it issues a request to do so and then receives packets transmitted to the specified multicast address. A host transmits packets to a multicast address without knowing which sites will receive them; the receivers have the responsibility of joining the group. To limit the scope of a multicast packet, a time-to-live (TTL) value is specified. The TTL is decremented each time it goes via a router. When the TTL is less than the link threshold, the packet is not forwarded. By convention, a TTL between 1 and 16 would be used to keep packets within a single network while a TTL of 127 would be specified for global traffic [Mbone].

2.3.2 Session Directory Protocol (SDP)

A session description expressed in SDP [SDP] is a first generation of conference control for Mbone based conferencing in the manner of a Session Directory – comparable to a TV program listing. SDP is a short structured textual description of the name and purpose of a conferencing session. SDP lists the media, protocols, codec formats, timing and transport information that are required to decide whether a session is likely to be of interest and to know how to start media tools to participate in the session.

SDP is purely a format for session description - it does not incorporate a transport protocol, and is intended to use different transport protocols as appropriate, including the Session Announcement Protocol (discussed in section 2.3.3), Session Initiation Protocol (section 2.3.4), Real-Time Streaming Protocol, electronic mail using the MIME extensions, and the Hyper-Text Transport Protocol (HTTP).



```
mjh 2890844526 2890942807 IN IP4 128.16.6.4
Multimedia seminar
Seminar on Internet multimedia
http://www.cs.ucl.ac.uk
mjh@isi.edu (mark handley)
IN IP4 224.2.17.12/127
t = 873397496 2873404696
Recvonly
Audio 49170 RTP/AVP 0
Application 32416 udp wb
Orient:portrait
Video 51372 RTP/AVP 31
```

Originator
Session ID and version timestamp
Originating host
Title
Information about the session
URL for more information
Email address to contact
Connection information (multicast address & TTL)
Start time (NTP timestamp)
Stop time (NTP timestamp)
Attribute indication session is receive only
PCM audio using RTP on port 49170
Wb application on port 32416
Wb is in portrait mode
H.261 video using RTP port 51372

Figure 11: Annotated SDP Session Description

In Figure 10 the session entitled ''Multimedia Seminar", will use audio, video and an application called ''wb", and that someone called ''Mark Handley" is responsible for the session.

However, this SDP also indicates the precise timing of the session (the ''t=" line gives start and stop times), that the session is multicast to group 224.2.17.12 with a TTL of 127, the audio is 8KHz law carried by RTP to UDP port 49170, the video is H.261 encoded, also over RTP but to port 51372, and the whiteboard program should be started up in portrait mode using port 32416.

Thus SDP includes the session name and a description of its purpose, the times the session is active, the media comprising the session, and information to receive those media (addresses, ports, formats and so on).

As resources necessary to participate in a session may be limited, some additional information may also be desirable, including information about the bandwidth to be used by the conference and contact information for the person responsible for the session.

In general, SDP must convey sufficient information to be able to join a session (with the possible exception of encryption keys) and to announce the resources to be used to non-participants that may need to know.

2.3.3 Session Announcement Protocol (SAP)

The Session Announcement Protocol (SAP) [Handley99] must be one of the simplest protocols around. To announce a multicast session, the session creator merely multicasts packets periodically to a well-known multicast group carrying an SDP description of the session that is going to take place. People that wish to know which sessions are going to be active simply listen to the same well-known multicast group, and receive those announcement packets. SAP and SDP are based on a concept where it is not possible to invite a certain user to a conference on demand.

2.3.4 Session Initiation Protocol (SIP)

The problem of not being able to invite users was solved by the second generation of conference control tools. The required functionality is provided by the Session Initiation Protocol SIP [SIP]. It is a simple, one step conference initiation protocol with basic support for address resolution, call forwarding and redirection. It is currently under development within the IETF MMUSIC (Multiparty Multimedia Session Control) working group. The protocol format is derived from SDP with elements from Hypertext Transfer Protocol (HTTP). SIP is a very lightweight signaling protocol with 6 methods and 20 header fields, mostly self-describing. Also it is meant to scale better (uses DNS) than signalling protocols designed in ITU.

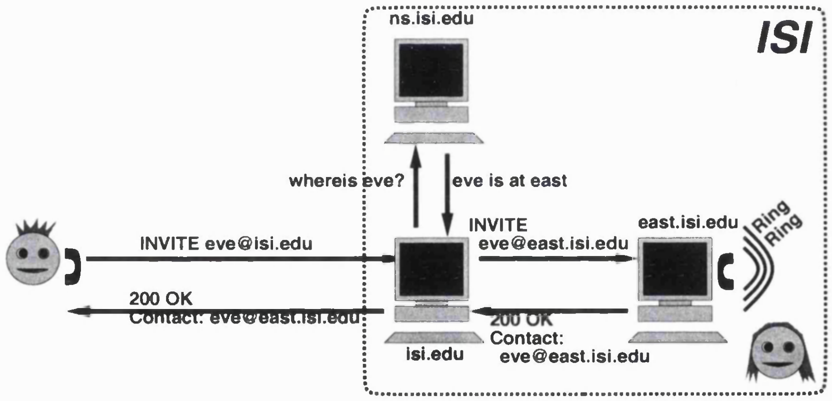


Figure 11: SIP Request Being Relayed

As illustrated in Figure 11, when eve@isi.edu is being called, an application (or phone exchange) looks up isi.edu in the Domain Name System (DNS) and looks for a SIP service record giving the address of the SIP server (in this case ns.isi.edu) for ISI. It then sends the SIP request to that machine. The server consults a database and discovers that Eve is logged onto east.isi site, and that the SIP server for Eve's department is east.isi.edu. It then sends the SIP request on to east.isi.edu, which again consults a database. This new database happens to be built dynamically by SIP clients registering when people log on. It turns out that Eve's workstation is not capable of multimedia conferencing. Instead east.isi.edu routes the call to her regular telephone and it acts as a gateway into the department PBX.

The example above uses proxies that relay the SIP call. Relaying is often performed when the gateway or firewall to a site wishes to hide any search that goes on internally from the caller's machine.

- **Summary**

The IETF model of conferencing supports large groups of humans and deliver real-time information at a number of levels. The functionality can be summarised as below:

- The Mbone based conferencing are multicast based
- It lacks explicit session membership and explicit conference control mechanisms
- These sessions consist of a number of many-to-many media streams supported using Real time Transport Protocol (RTP) [RTP] and Real time Transport Control Protocol (RTCP) using IP multicast. The only conference control information available during the course of a session is that distributed in the RTCP session information, i.e. an approximate membership list with some attributes per member [RTP].
- In traditional Internet, network resources are not explicitly reserved which may be required for real time delivery of audio and video for the duration of a conference. In IETF new service model and protocols (such as RSVP) to reserve capacity is defined which is more flexible than that available with circuit switching.
- Distributed architecture to advertise sessions (SAP) and invite participants in a conference (SIP) which are "light weight" protocols

Figure 12 depicts a summary of most of the protocols discussed above for Mbone based Internet conferencing.

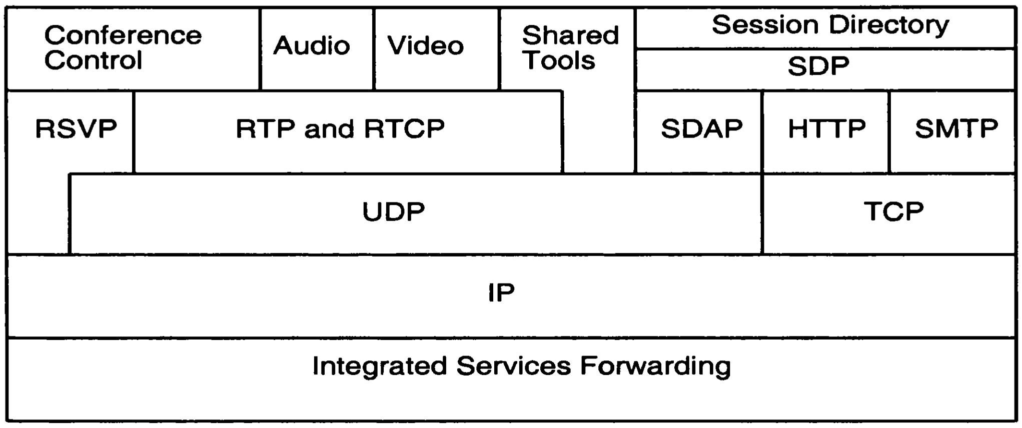


Figure 12: Internet conferencing protocol stack

2.5 Other conferencing systems

In Mbone based conferencing there are no actual conference control tool that will co-ordinate and manage users and applications or meet other criteria discussed in chapter 1 (section 1.3) for conference control. This is because, IETF based conferencing was designed for broadcasting IETF meetings over the Internet using "best-effort" quality targetting the research and the academic community, where strict control is not required. However, as the popularity of the Internet grows, the number of users of Mbone based conferencing increases with it and today users of conferencing applications demand comfortable session initiation and session control functionality. Also Mbone based conferencing does not provide any mechanisms for ensuring QoS in conferencing. Therefore, a number of independent conferencing architectures and control mechanisms have been developed outside the IETF and the ITU standard bodies. Some of these systems will be briefly mentioned here:

2.5.1 The CAR conferencing systems

Around 1990 – 1991 when H.320 was being developed, the European RACE CAR project was developing a multi-party multimedia conferencing system. CAR used ISDN channels, with one channel for audio/video data and the second ISDN one for IP data. A centralised conference control ran over this secondary channel permitting users to join and leave the session, video switch, floor control, and introduce control of shared applications. These applications included existing X-window based tools shared using Shared-X and a shared sketchpad which used the TCP based protocol.

CAR was more advanced than H.320 because CAR used remote procedure call (RPC) to connect components together while H.320 uses ASN.1. RPC has some attractive infrastructure support (e.g. dynamic binding and better language integration) but as mentioned the actual audio and video communications are done using ISDN. So it suffered badly from unstable nature of ISDN services and terminals available at that time. In particular, its centralised architecture, and its use of a TCP-based RPC mechanisms did not deal with failure that well.

Shared-X also taught a lesson in how not to design a collaboration mechanism. It consisted of a "bridge" which acted as an X server to the client application to be shared and mapped all the X resources and identifiers for connections to multiple real X-servers. In addition to causing traffic concentration, this centralised model attempts to keep all X-servers in step. Shared-X coped with the failure of a site, it locked up all participants displays for around 90 seconds whilst it timed out the failed connection [Handley M].

2.5.2 The LAKES architecture

The Lakes architecture has been developed at the IBM research laboratories as an "architecture for collaborative networking" [IBM94]. The goal of the Lakes architecture is to provide multimedia group communication services independent of underlying networks and specific personal computer platforms.

The Lakes architecture defines a platform that offers a uniform API towards collaborative applications. The focus is not specifically on teleconferencing but more general on setup and handling of information stream(s) between groups of application entities and/or hardware devices. The functionality for (real-time) stream management include:

- Call control between groups of application entities, i.e. setup and teardown of communication
- Provision of unidirectional transmission facilities (channels) from a single source to one or more destinations with application-defined transmission requirements, such as QoS, signal type (analogue or digital), data class etc.
- Synchronisation and resource sharing by means of (global and as well as per-application) tokens
- Maintenance of node information in a Lakes network in an address book.

The services offered by the Lakes platform may be categorised into:

- application services
- synchronisation services
- transport layer services
- call manager services – name resolution, placing outgoing calls and accepting incoming calls and implementation of policies for these two functions.

The LAKES Architecture was heavily influenced by ITU's MCU design topology. In order to join a LAKES architecture of conferencing, a machines needs to have certain software installed in it and hardware changes made to it. This could be an expensive process for the users.

2.6 Conference Controls

Conference control has some basic system requirements (discussed in chapter 1, section 1.3) but it has several other usages. It can be used for:

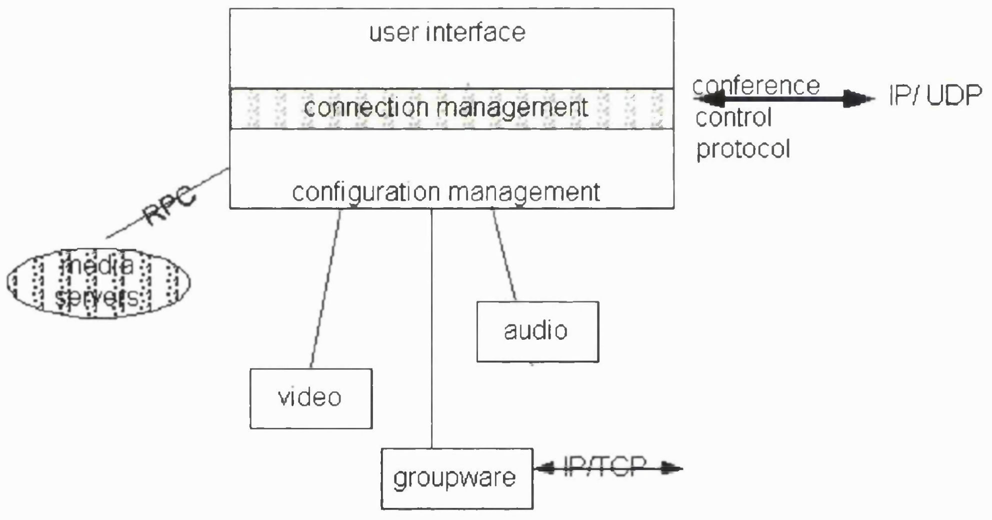
- Address book and Session listing - Looking up addresses of users and directory listing service like LDAP and SDP
- Negotiation – Among other conference control functions, one of the main uses of ITU's H.245 and T.124 is to negotiate and exchange capabilities of end systems

- Media control – IETF's Message Bus [Ott/Perkins] is used as a way to synchronise locally situated applications. CONFCTRL is used as a remote controller to set and change audio volumes, video rate etc. in remote locations.
- Address resolution – H.323's gatekeeper provides address translation from one format to the other. So for example, if a user enters an email address of a user that can be translated as the user's IP or machine address.
- Interoperability – A conference controller can be used to provide interoperability between incompatible systems. This thesis looks at details of such usage.
- Security and charging – The services provided by a conference needs to be charged for. Hence security mechanisms are required to ensure the correct user is billed for and the charging mechanism needs to be an attractive one for users. In chapter 6, such a charging model is analysed although the security requirements are not discussed in this thesis.
- Intelligent Network function – A conference controller can provide functions like Call Waiting when a user is busy, Call Forwarding, Call Logging, Malicious Call Identification and Call Transfer among others. Confman 2.0 [Fromme] looked at some of these functions although currently no conference control tools provide IN functions.
- Resource management – A conference controller can be used to manage bandwidth or distribute calls to different servers depending on the load on various servers. H.323 gatekeeper is a prime example of such usage.

Below, a list of conference control mechanisms are briefly discussed. These conference control systems provide one or more of the services listed above. Most of these conference control systems were researched as a compliment of IETF's model of conferencing. In other words, systems like MMCC, Confman 2.0 and CCCP (discussed below) all came into being because the IETF model of conferencing did not provide a mechanism for co-ordinating and managing users or applications, address resolution or media control. Therefore, the related work discussed below, are complementary to SIP/SAP and SDR and they have been experimented with Mbone tools like vic, vat, rat etc.

2.6.1 MMCC

Multimedia Conference Control Program (MMCC)[Schooler91] from USC/ISI was developed as a prototype session orchestration tool for point-to-point and multipoint multimedia conference. Its main functions are to supply *session creation* and *maintenance services* and to *manage media.* The media tools are separate application and can include vat, nv, and wb. Built on a distributed, peer-to-peer model, MMCC is meant to execute continuously on a workstation residing at each conference site. The media tools may be executed on a separate machine. Conference participation is by invitation only.



Coordinated Management of Separate services

Figure 13: Coordinated Management of separate services

To establish a conference, a user enters the required information and invites other users to participate. MMCC allows a caller to explicitly invite others to join a conference and alerts them to accept or reject the invitation. When a user creates a session, MMCC assigns a conference address and identifier, spawns the selected media tools with the specified configuration (e.g. device, bandwidth, and coding algorithm) if it can be met, and alerts the callee to accept or decline invitations.

When a participant leaves the conference, MMCC tears down the media channels. MMCC allows limited remote control of data rate and hardware devices (e.g. cameras, codecs, and monitors). Although the actual data flow is left to the individual media components, MMCC passes session and control information to them and to other MMCCs as shown in Figure 13. This information includes lists of participants and timing information for inter-media synchronisation. MMCCs use the Connection Control Protocol (CCP) to communicate with each other. Multicasting of control information is via sequential unicast rather than via the Mbone.

MMCC was designed to be an effective tool for conference control over the Mbone. However, it did not deal with issues like security or other conferencing tools that do not run over the Mbone.

2.6.2 CCCP

The Conference Control Channel Protocol (CCCP) [Handley] is used to bind different conference components together that are otherwise stand-alone on the Mbone as *a common communication channel.* This offers facilities and services for the applications to send to each other. This is akin to the interprocess communication facilities offered by the operating system. CCCP runs on a bus model, passing all messages around a single multicast group per conference. The CCC naming scheme is based upon the attributes of an application that could be used in deciding whether to receive a message. Separate applications need to use some regular expressions (base types) to ensure that common applications can communicate with each other. The following have been suggested for this purpose:

- Audio.send – the application needs information about who is sending audio
- Video.send – the application needs information about who is sending video
- Video.recv – the application wants to receive video

- Session.remote – the application is interested in knowing the existence of remote applications
- Floor.manager – the application is a floor manager

There are different levels of reliability and ordering in CCCP. The main constraint for CCCP is that it can communicate with different Mbone based applications but it does not specify how it should deal with applications that are not capable of running on top of IP multicast.

2.6.2 CONFCNTLR

Confcntlr [Perry] is a Mbone tool that works in conjunction with other videoconference tools to enhance the usability of video and audio. Mbone tools like vic and vat have separate interfaces and none can be controlled remotely. Also most of the tools do not allow a user at one host to launch or stop the video conference tools on another host or to change the parameters on the remote host once the tools have started. Confctrl is designed to overcome these shortcomings by allowing local and remote control of the audio and video tools and provides a single interface to vic and vat. Users at one site can start a video or audio session at another site and can specify the parameter values. Once vic or vat is running, remote users can change the audio or video settings without stopping and restarting tools and they can turn video transmission on or off without stopping and restarting vic.

2.6.3 SCCP

Simple Conference Control Protocol for tightly coupled conferencing is based on the Internet Multimedia Conferencing Architecture [Ott J (a)]. SCCP was designed as a proposal for a model of tightly coupled conferencing over the Mbone. The functions provided are based on the services offered by the ITU-T recommendations of the T.120 and H.320 series. The aim of SCCP is to define a simple conference control protocol that is rich enough to allow construction of gateway to the T.120 world. Also the protocol is intended to be robust to failures. *In contrast to the ITU-T recommendations, it allows the use of IP multicast both for multimedia information and for other applications and control streams.* The main functions of SCCP are:

- Management of the set of members participating in the conference
- Management of the set of applications/media that constitute the conference
- Floor control
- Assignment of a special "conductor" role to one participant

Conferences that are controlled by SCCP may be convened using an invitation protocol such as SIP.

SCCP is based on the idea of maintaining a common state for the entire conference, the conference context, at all participants. This state is partitioned into objects. SCCP distinguishes four types of objects:

- variables, which are general repositories for global state that does not fit elsewhere, such as conference policy;
- sessions, which are a special kind of variable that represent global state about an application session;
- tokens, which are a special kind of variable that allow synchronisation operations; and
- members, which represent the information specific to a single member.

The design concepts of SCCP are very efficient and a lot of it is very relevant for the CCCS. However, SCCP especially has been designed as a conference controller for tightly coupled conferencing and not for all types of conferencing. For scalability, the "conductor" or the "receptionist" role of SCCP can be a conflicting concept. For example, when a user wants to join a conference it must send a JOIN request to the group and it is up to the receptionist for answering the JOIN request, unless this user happens to be the first participant in this group. The receptionist role cannot be duplicated without proper handover, in other words only one receptionist can pass its information to another (central point of failure). Therefore, if the receptionist host's link is down and the role hasn't been transferred, the new participant of a conference may never receive a confirmation for his/her JOIN request.

2.6.4 Confman

Confman [Fromme] allows easy setup and handling of multimedia conferences (especially Mbone based) over the Internet. The Confman core system is divided in subsystems which support different services. The conference control is one of these services. The functionality of conference control services include initiation of conference, an address book facility which stores addresses of conference members, a session directory that contains the Mbone broadcast announcements like SAP and some basic telephone services like call waiting and call forwarding.

The Confman system is divided in two parts: the core system and the user interface. The system uses its own communication protocol, called Confman Conference Control Protocol (CCCP). This protocol handles all data exchanges that take place when starting and stopping of conference medias, joining and leaving of conference members, and the start and termination of a conference. The core system contains all core functionality parts of the Confman system such as protocol handling, address book management, conference management and tool control. The user interface contains code for the graphical user interface and for user intercation.

Both parts of the system are interconnected by object-oriented middleware called Remote Object Invocation (ROI). As the ROI uses the TCP, it allows scenarios where the Confman user interface is loaded from a WWW server and run in a browser. The user interface will be running in a Java interpreter and will be connected to the core component via Confman proxy by ROI as shown in Figure below:

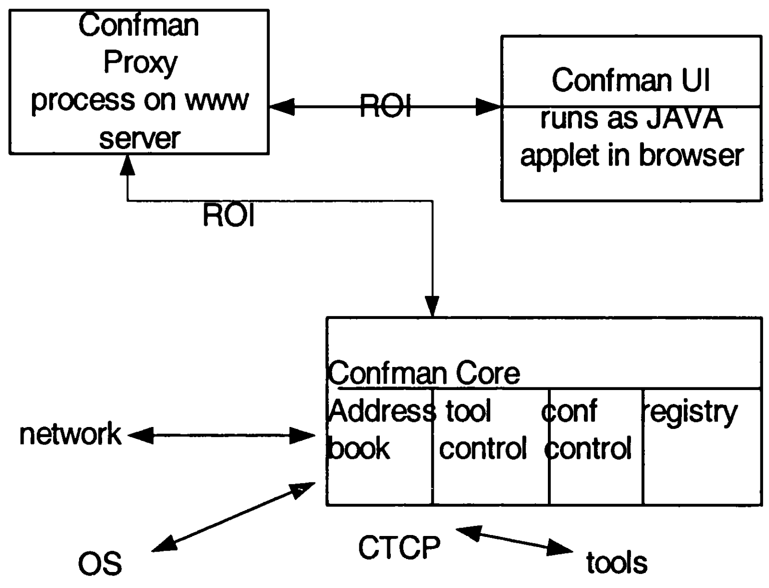


Figure 14: Confman running in a WWW server

As shown in Figure 14, the core system can be divided in four main parts:

- The *Registry* component starts and initialises the system and it holds configuration parameters.
- The *Address book* component allows access to directory servers and to local file-based address books. Directory servers are accesses by using the Lightweight Directory Access Protocol (LDAP).
- The major component is the *Conference Control* part. It contains address resolution, session initiation and conference control protocols allowing information exchange with other conference-control systems for conference setup and protocol.
- The fourth part is the *Tool control* component. It handles the start and control of media tools such as audio or video transmission components by using Confman Tool Control Protocol (CTCP). CTCP uses local scope multicast to communicate with the tools its functionality is similar to the conference bus in vic.

The architecture and services provided by Confman are based on object oriented design and it has a high potential for further developments. The design is very flexible and a system like this can provide useful input for future developments of CCCS which will be discussed in the next chapter.

2.7 Gateways

Recently, there has been a great interest in protocols and procedures for interworking between different conferencing protocols. This is particularly important as conference models move from circuit switched to packet switched environments. Among these, the SGCP and the SS7 ISUP gateways are discussed below because the specification of these gateways provide useful and relevant input for designing CCCS's gateway functions :

2.7.1 H.323 and SS7 ISUP Gateway

Currently there is work in progress to interwork conventional telephones with PC based conferencing. These conferences are H.323 based and the SS7 ISUP protocol is used in circuit based networks. One of the most important differences between packet based IP network and circuit based network is that there is no actual channel, such as ISDN B-channel or PSTN trunk circuit, in the IP network. In other words, the H.323 terminals can have more "channels"; logical channels to carry more than one application such as audio, video and data in one call. Because of this difference, the call control specified in ITU's H.323 is more complicated [Ma]. In circuit-based network, the call signaling (i.e., setup and teardown) and bearer capability control are processed in one protocol, Q.931 for ISDN and ISUP for SS7 network. However, since different H.323 terminals may have different logical channel capabilities, ITU's H.323 has specified the call signaling and bearer capability control in two separate protocols, H.225 and H.245.

Interworking between SS7 ISUP and H.323 (actually H.225 and H.245) is quite straightforward. For a signaling gateway between packet based network and circuit network, the information from the bearer capability information element in the H.225 SETUP message is enough to set up a call in the ISDN or PSTN through SS7 ISUP. H.245 is just used for logical channel setup based on bearer capability carried in the

H.225 SETUP message. The H.323-SS7 gateway is specifically designed for interworking the PSTN and the IP based network. Although this gateway performs its job as a signaling gateway quite adequately and sets up a voice connection between a PC and a telephone, the call control elements that are required in a PC based conference are absent from it.

2.7.2 Simple Gateway Control Protocol (SGCP)

It is a protocol for controlling voice over IP (VoIP) gateways from external call control elements. The SGCP assumes a call control architecture where the call control "intelligence" is outside the gateways and handled by an external entity. In the SGCP model, the VoIP gateways focus on the audio signal translation from circuit switched networks to IP, while a call agent handles the call signalling and call processing functions. SGCP will be used to control and process several calls coming through different H.323 and SS7 ISUP gateways discussed above for example; in other words, SGCP defines connection and endpoint handling of a VoIP gateway.

The services of SGCP consist of connection and endpoint handling commands for a gateway. It instructs a gateway to watch for specific events such as DTMF busy tone on a special endpoint, answerphone etc. In order to implement proper call signalling, the SGCP must keep track of the state of the gateways, and the gateway must make sure that events are properly reported to the call agent. An endpoint may need to provide necessary authentication to the local call agent in order to communicate with a gateway or send audio or data on the network. The Call Agent uses the SGCP to provision the gateways with the description of connection parameters such as IP addresses, UDP port and RTP profiles [SGCP].

SGCP messages are transmitted over UDP. Commands are sent to one of the IP addresses defined in the DNS for the specified endpoint. The responses are sent back to the source address of the commands. SGCP messages being carried over UDP, may be subject to losses. In the absence of a timely response, commands are repeated. SGCP entities are expected to keep in memory a list of responses that they sent to the recent transaction, i.e. a list of all the responses they sent over the last 30 seconds, and a list of the transactions that are currently being executed. The transaction identifiers of incoming commands are compared to the transaction identifiers of the recent responses. If a match is found, the SGCP entity does not execute the transaction, but simply repeats the response. The retransmission timer values are typically network dependant. Although SGCP contained some good design features of a gateway control protocol, the SGCP does not deal with race conditions very well. Also, the security features needed to be enhanced.

2.8 Conclusion

In this chapter, different models of conferencing, conference control and gateways have been reviewed. We observed that H.323 is aimed (along with all the other H series) at extending the ISDN videoconferencing standards to unreliable packet LANs, but it is not aimed at solving the scaling problems of WANs. These sort of conferencing over the Internet has a large amount of baggage associated with it. The nature of call setup messages and control of multiplexing show that it is not for conferencing but telephony. However, there is little reason to expect a direct derivative of a standard like H.323 that was introduced in 1990, standardised with the design of special purpose hardware for videoconferencing over circuit-switched telephone networks, to be a good guide to the 1999 design of software that runs on general purpose desktop computers connected by a best-effort packet-switched data network. The contexts are fundamentally different. Nevertheless, there is a huge support for H.323 for providing a complete solution for video, data and audio in a monolithic package. These sets of standards will remain in the market for a distant future.

On the other hand, the IETF's model of conferencing provides scalability and it is especially suited to work over the Internet. However, this standard body does not provide a conference control by which users could apply a floor control mechanism or provide charging and accounting information for the usage of a conference if they wish to.

Therefore it can be concluded that, currently, the Internet based conferencing is lacking a conference control function and the ITU based conferencing seems to be a “heavy weight” protocol that does not scale effectively over the Internet. Therefore, a compromise is required for a “control protocol” that is designed to be distributed over different networks but at the same time it can control and coordinate the activities of multiple users and applications. Also, as pointed out in section 2.6, one of the uses of conference control is to be able to support and interwork different protocols which is currently not provided in any system. One way to unify different protocols is to interwork the existing protocols and combine the main call control functions that are available in all of them. However, it is impossible to design a true generic conference protocol that can unify all conferencing architectures ever designed. So, an attempt to gather the main design principles of most commonly available conferencing architectures is done in the next chapter.

Chapter 3

Common Conference Control Services (CCCS)

This chapter is divided into three main parts. The first part describes a framework that provides the basis for group communication in multimedia conferencing systems. The second part describes conference control and its position in that framework. The final part describes the Common Conference Control Services (CCCS), its design, operation and performance.

Conference control which is an integral part of conferencing, includes two types of services: user visible services, and internal management services. User visible services are user invoked and range from user interaction to application interaction, while other services are for internal management of a conference. The latter include services like media control (discussed in chapter 2), consistency and interoperability between different architectures. These architectures could be either explicit control based conferencing normally seen in ITU standards or implicit control, as with conferencing systems dominating IETF's standards. The purpose of the CCCS is to enable the interaction of different conferencing systems and allow them to be integrated into one complete architecture.

CCCS's implementation shows a proof of concept of generic conference control model. In this chapter, the details of this implementation and the design behind this model are provided. The CCCS implementation acts as a conference control gateway between two prominent protocols, H.323 and SIP. The main design principle of this framework is to let the common activities that are present in a canonical model of conferencing interact. The objective is to be extensible so that other conference control protocols can interwork with IETF and ITU standardised protocols. It allows the call set up between different protocols to take place successfully and then allow other user visible services to be invoked. Therefore, if a conference policy enforces users to follow a strict floor control model for example, this design will be able to cope with that. The framework works over IP multicast and Unicast.

3.1 The Basis of Group Communication: A conference protocol stack

Various conference control communication services discussed throughout this thesis can be represented in a conference protocol stack. This stack can be considered as a design model which can be used as a basis of a generic framework for a conferencing system. It allows the design of a complete solution for a conferencing system, as well as it enables one to concentrate on specific details if required. The basis of group communication for conferencing, the conference protocol stack follows a layered architectural pattern which is suitable for designing a system whose dominant characteristic is a mix of low and high level issues. In this pattern high level operations rely on the lower level ones.

This generic architecture can be divided into three main sublayers shown in Figure 16. The top layer is divided into two segments: a) conference management and b) groupware applications. The CCCS discussed in this chapter mainly falls under the category of conference management. Both applications and management on the top layer of the stack require services from the Session and Resource Management. The charging model discussed in Chapter 6 is used alongside session management for a conference. The Resource Management part performs resource allocation such as bandwidth and delay (beyond the scope of this thesis). The third layer provides communication services provided by the network itself. As shown in chapter 4 and chapter 5, network services can be adapted to meet the requirements of conference management. Examining individual layers in more detail reveals that they are complex entities consisting of different components. Components in each layer need to interact with each other. So for example, the applications need to cooperate with the conference control tasks.

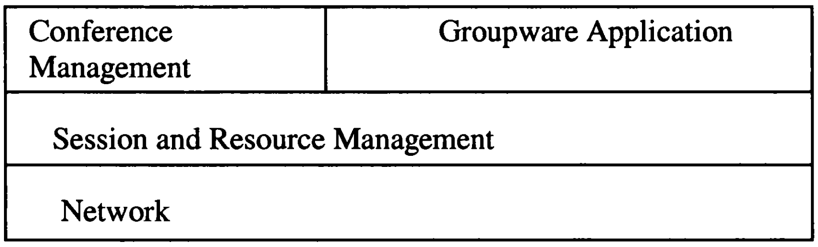


Figure 15: Conference protocol stack

3.1.1 Components of the architecture

Ott et al [Ott J] presented a Multipoint Communication Layer (MCL) which is a communication platform that aims at supporting Groupware applications and integrating them into a desktop multimedia system. MCL's design architecture shows a similar pattern to that presented in Figure 15.

The components in Figure 15 show the following:

a) Conference Management: The conference management deals with all functions related to co-ordination and management of a conference. These functions could be visible or invisible to users and are the set of functions that facilitate the user's requirements (discussed in chapter 1, section 1.3 and 2.6).
b) Groupware application: This represents a set of separate applications that can be used to convey audio, video (such as vic, rat, vat) or a monolithic conferencing application like CUSeeMe.
c) Session and Resource Management: The Session and Resource Management cover generic transport and synchronization mechanisms. This comprises a set of protocols that can guarantee the QoS by reserving resources such as RSVP [RSVP] for the Internet or Q.2931 for ATM. This layer can also be used to conceal most network specifics from the transport service user for a given transport connection. For example, error control can be adjusted, flow control added etc. so that a high quality level can be demanded at the transport service interface. The synchronization

functionality of this layer can be used to synchronize different servers that are managing different conference control services at different locations.

d) Network layer: The underlying network can be private, PSTN or the IP based Internet.

In the following sections, the conference management part of the architecture is expanded on; it deals with the analysis, design, requirements and implementation issues of this layer.

3.1.2 Conference administration services

Conference control has two different types of interactions:

- Inter-system communication occurs between peer entities running on different systems: the conference management entities communicate with one another using a conference management protocol over the network. The set of application entities exchange information within application sessions. These application sessions are isolated from one another and they may or may not be controlled by the conference management entity. The peer applications may need to have compatible attributes, for example, they will need the same codecs.
- Intra-system communication is used to coordinate the otherwise unrelated local groupware applications on each teleconferencing systems and integrate them with the conference management entity as well as to provide access to the conference control services. Examples of intra system communication system includes systems like mbus [Ott/Perkins] or LBL's [McCanne] message bus.

Figure 16 is an example of a system model that shows the interactions between two teleconferencing systems in a point-to-point conference, the various protocols in use and CCCS's position in this model. The components can be described as below:

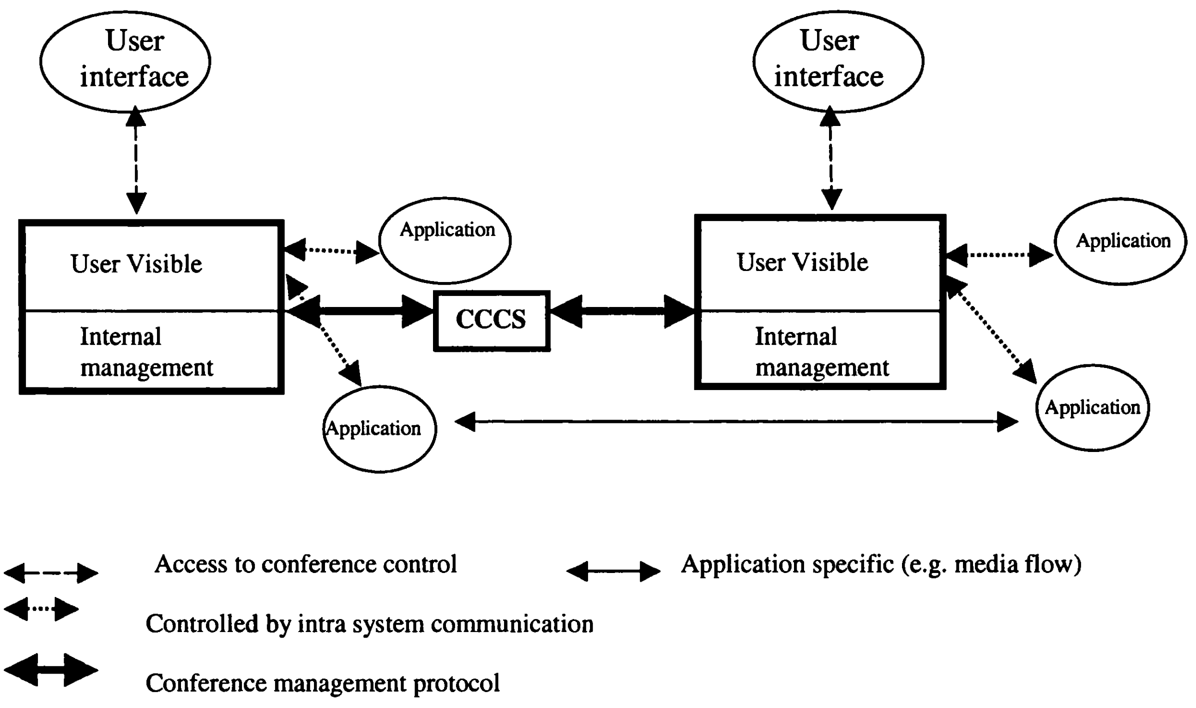


Figure 16: Components for conference management services

Users access different types of visible services, such as inviting another user, or joining a conference (discussed in section 3.2) via an Application Programming Interface (API) or a "User interface". The local applications in one system are co-ordinated by its intra system communication method. The CCCS acts as inter system communication framework between different models. The services provided by the CCCS is broken into two main parts: the user visible part referred as MCS (Main Control Services) and Gateway Services (GS). The MCS is responsible for membership control, floor control and authentication. If one user is inviting another user who happens to be in a different type of conference, the internal management functions of CCCS, the GS part performs the interoperation. Also the visible part of the CCCS, Main Control Services (MCS) supplies connection management in the form of session establishment, maintenance and disconnection. The CCCS performs these tasks remotely with peer connection management entities in remote units using a connection control protocol (TCP or UDP over IP as underlying transport protocol).

3.2 CCCS's user visible services

The user visible services of a conference control service are divided into different functional groups as shown in Table 4. These are: conference configuration, participation management, floor control and security [Ott J].

- Conference configuration: Conference Configuration essentially refers to defining a conference profile. Means for specifying and enforcing conference policies are also provided. Specifying conference name, number or list of participants, required resources and access modes such as users password, joining fee etc. are part of this service. A conference profile can also define permissible participants, available roles (e.g. chair, speaker) and associated permissions. The role assignment by either a system or a user follows: a) assign role initially or b) request role and then grant/deny/share role.
- Participation management: Participation management comprises services for setting up conferences, point-to-point calls, group or individual invitation, and termination. Furthermore, functions for charging the membership of a conference are included. Participants may join or leave a conference on their own, or they may be invited or excluded from a conference. Changing from one conference to another is included in this function as well.
- Floor control: Floor control is a metaphor for "assigning the floor to a speaker", which is applicable to any kind of sharable resource within conferencing and collaboration environments [Dommel]. A floor is an individual temporary access or manipulation permission for a specific shared resource, e.g., a telepointer or voice-channel, allowing for concurrent and conflict-free resource access by several conferees. There are several types of floor control policy available for use by collaborative environments [Greenburg 91]:
  - Free floor – concurrency control is mediated through user (social) protocols.
  - The pre-emptive scheme – allows any recent requestor to take floor.
  - Explicit release – the floor holder must relinquish control before anyone else can claim access.

- Round-robin scheme – this gives each participant a quantum of time during which they have full access rights.
- The central moderator – the chair has the right to assign floor control to others.
- The pause detection – during a specific time period if one user does not use the access rights, then control is removed.

- Security functions: Security functions are value added options for conference control and are a part of participation management as well as conference configuration. Authentication is performed when a participant enters the conference and may be repeated arbitrarily during the conference course.

Functional group	Conference control services semantics	
Conference configuration	Profile definition	-Define permissible participants -Billing/charging reqs -Available roles and associated permission
	Role assignment	-Assign role initially -request role -deny/grant/share role
Participation management	Join	Join a conference
	Leave	Leave a conference
	Invite	-Invite a participant -invite a group
	Exclude	-Exclude participants as part of conf termination
Floor control	Floor assignment	-Assign floor initially -request floor -grant/deny floor -give up floor
Security	Authentication	-check password upon joining -distribution of session key

Table 4: Summary of user-visible services of a conference control [Ott J]

APIs facilitates the user visible functions mentioned above in a multimedia conference. For example, when a person wishes to invite another user, he enters the "invite" or "call" button with necessary details of the invitee. Upon receiving an invite, the callee either accepts or rejects the invitation or enters a password if required. The MCS (Main Control Services) part of the CCCS facilitates participation management, floor control (FIFO scheme), conference profile identification, some restricted security services and basic error controls. Using these API as part of MCS, a participant can join or leave a conference, invite another participant, check conference profile and grant/refuse floor request.

When a conference is created it can be advertised. The MCS does not perform the advertising itself because different conferencing applications have their own way of advertising conferences like SDR [SDP] or LDAP [LDAP]. The job of the MCS starts when the participants are trying to invite another participant. If the invitee is already logged on, an invitation message appears on their screen. Otherwise, MCS returns a message indicating they are not contactable. If the participants have a conferencing application loaded on their machine (which could be H.323 compliant or Mbone[Mbone] based application) the invitation message or the equivalent of telephone "ring" appears on their machine in the format that is specified within that particular architecture. If the initiator/caller is using an architecture that is different to callee's, the Gateway Services (GS) translates those call control functions (CCCS's gateway functionality is discussed in details in section 3.3.5).

3.3 Internal management

The user-visible conference management services described in the previous section provide functionality to control the source of a teleconference and reflect the participants' behaviour. Normally users carry out these functions using a conferencing/ groupware application. These groupware applications are considered independent entities rather than merged with a conference control tool. Currently most of the Mbone based conferencing applications – audio, video communication tools as well as signalling operate in a stand-alone manner, i.e. without a conference management entity.

It is the task of the internal management services of conference control to integrate the different types of conference management entities in order to make them appear as coherent teleconferencing system [Schooler93]. As discussed in 3.2.1, inter-system communication and intra-system communication are two ways to solve the synchronisation and integration aspects of conferencing. In order for the intersystem communication to accomplish the integration and provide a richer set of services, it must perform: a) interoperability b) consistency.

Interoperability means that users from two or more different application systems can collaborate. Regardless of the supplier of the application, if there are a number of tools and media available to the users they should be able to negotiate and determine a common set of tools to exchange information. The level of interoperability can vary. For example, user A from system 1 can only receive and send audio, whereas user B from system 2 can receive audio and video. So there must be a capability exchange mechanism to find out if they can both at least send and receive audio. Therefore, it can be said that there are mainly two types of gateway involved in this situation : a) the call control or signalling gateway b) the media gateway. The call control gateway maps the call control functions (e.g. SIP and H.225) from one client to the other to set up a call whereas a media gateway performs different types of codec conversion for different media.

Consistency provides a way to report a list of different types of applications, participants, and their status. A conference could be in paused (i.e. people are on break), closed or at the beginning stage. As the number of participants scale to thousands over the Internet, it becomes very difficult to get an exact list of all the participants and their status. Therefore, it may be possible to get the status of a number of participants at one time from one link which may not be consistent with the number appearing to another link at the same time. There are ways to address the problem [Schulzrinne/Rosenberg INFOCOM] like statistical sampling or a timer backoff algorithm for example. The section below discusses the issues related to CCCS's interoperability functions.

3.3.1 Design model of the CCCS's Interoperability functions

The design approach of the CCCS's main internal management function, interoperability follows a well known software engineering design pattern. The development is based around the ***BROKER*** (also referred as ***DISPATCHER)*** architectural pattern. The Broker architectural pattern can be used to structure distributed software systems with decoupled components that interact by remote service invocations [Buschman]. A Broker component is responsible for coordinating communication, such as forwarding requests, as well as for transmitting results and exceptions. This type of architectural pattern is especially suitable where the environment is a distributed and possibly a heterogeneous system with independent cooperating components.

A Broker component introduces better decoupling of clients and servers. Servers or clients register themselves with the Broker, and make their services available to clients through Application Programming Interfaces (API). Clients access the functionality of servers or other clients by sending requests via the Broker. For example, a H.323 client can access the services a SIP proxy server provides via CCCS. So if a H.323 client wants to contact a client who has a set up where the calls are redirected to his/her home PC, the H.323 client's request is processed by a SIP redirect server. However, the H.323 client contacted the SIP redirect server via CCCS without knowing the details of the redirect server's operations. Therefore, in this case CCCS *is the BROKER or the DISPATCHER.*

By using the Broker pattern, an application can access distributed services simply by sending message calls to the appropriate object, instead of focusing on low-level inter-process communication. In addition, the Broker architecture is flexible, in that it allows dynamic change, addition, deletion, and relocation of objects.

The Broker system offers a path to the integration of two core technologies: distribution and object technology. They also extend object models from single applications to distributed applications consisting of decoupled components that run on heterogeneous machines and that can be written in different programming languages.

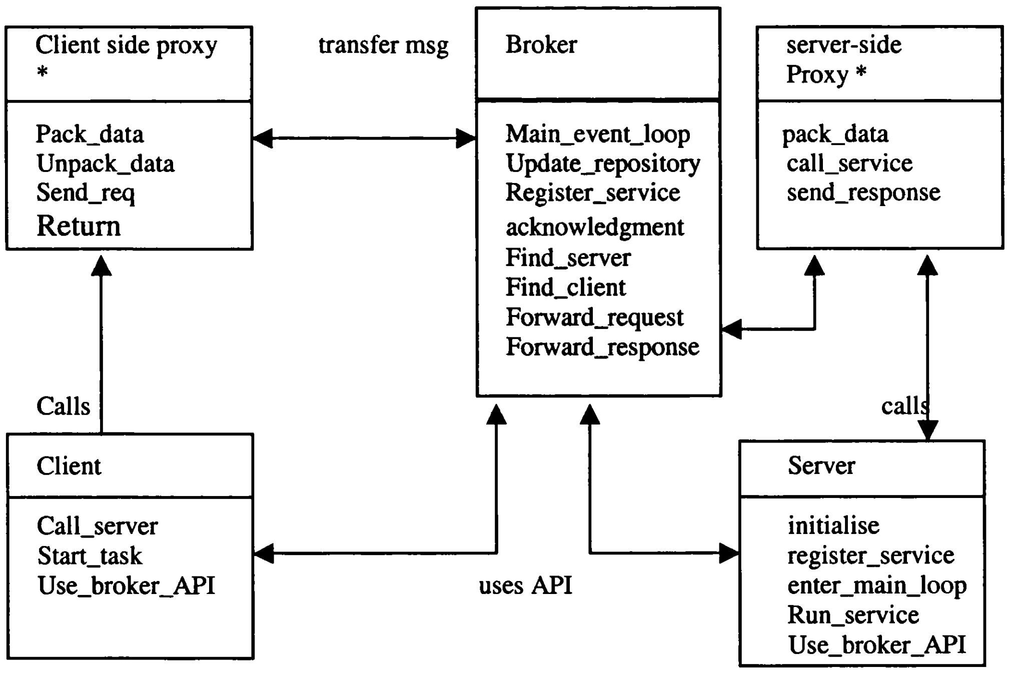
Structure: the Broker architectural pattern comprises six types of participating components: clients, servers, brokers, bridges, client-side proxies and server-side proxies. Figure 17 shows a typical Broker architecture's responsibility and the participating components. Among these components, this system implemented mainly three types: clients, servers and brokers. In the context of the Broker pattern, the clients are the available Conferencing Applications (independent of their underlying architectural stack). They can be scattered on different types of networks and they can be connected to the Internet either using a gateway or rely on Internet providers to offer connectivity. When they need to connect to each other they do so using CCCS's call control and signalling functions. If a network comprises of lot of different clients and servers, either one of the client or the server or proxy as shown in Figure 18 may connect to CCCS and it acts as a Broker to register the entities and transfer messages between two or more incompatible entities. Depending on the requirements of the whole system, additional services – such as naming services can be integrated into the Broker. Name services provide association between names and objects. To resolve a name, a name service determines which server is associated with a given name.

Broker	Collaborators
Responsibility - (Un) Register servers - transfers messages - offers API - error recovery - locates servers/clients	- client - server - client-side proxy - server-side proxy - Bridge

Figure 17: Responsibility and collaborators of a Broker

CCCS, which is an example of a Broker system, is involved in forwarding requests and responses from different clients. It is also up to the Broker to find the location of appropriate server or a client that has been requested for. The dynamics of Figure 18 can be described as below:

- The Broker (in this case the CCCS) is started in the initialisation phase of the system. The broker enters its event loop and waits for incoming messages.
- The user, or some other entity, starts a server application. First, the server executes its initialisation code. After initialisation is complete, the server registers itself with the Broker.
- The Broker receives the incoming registration request from the server. It extracts all necessary information (see section 3.4 for a list of information required for this type of operation) and stores it into one or more repositories. These repositories are used to locate and activate servers. An acknowledgment is sent back to the requesting servers after registration is complete.
- After receiving the acknowledgement from the Broker, the server enters its main loop waiting for incoming client requests.



* optional/ not discussed in this thesis

☐ Methods used in CCCS as part of architectural design

Figure 18: Objects involved in a Broker system

3.3.2 Main operation of the CCCS

The CCCS follows a number of steps to accomplish its function as gateway and a conference control provider. The functionality of the CCCS as a Broker can be divided into three categories: a) initialisation and registry update b) client registration and c) session management

-

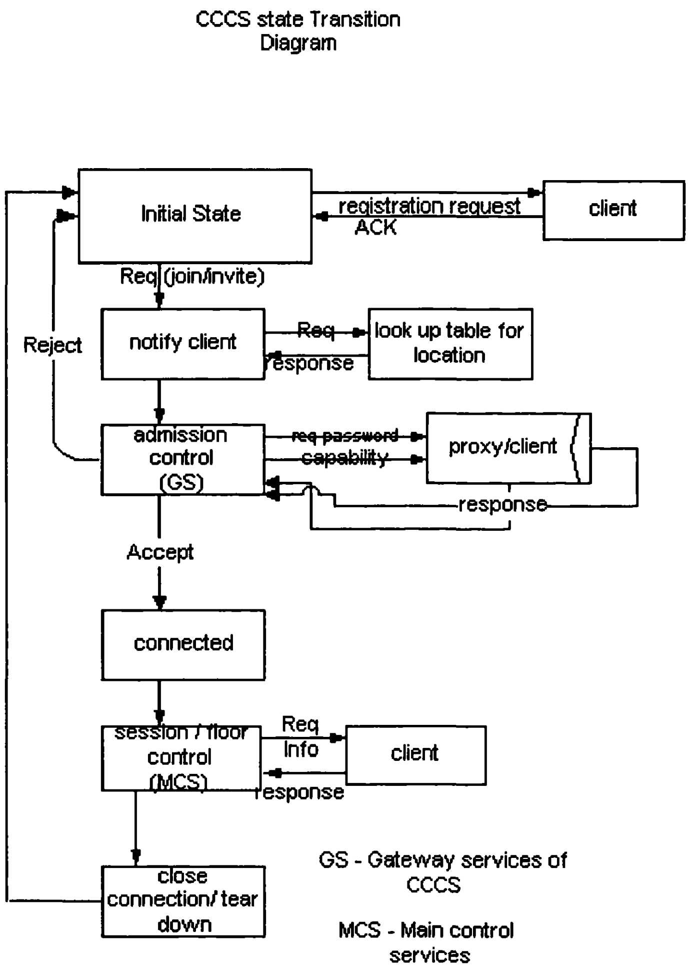


Figure 19: State Transition diagram of CCCS

<u>Initialisation and registry update</u>

The CCCS is initialised to process messages. When a client processes a CONNECT [Stevens (a)], (the socket system call to establish a connection with the server), GS part of CCCS updates its registry. The registry must keep a track of the following:

- Protocol type - the protocol types of conferences (e.g H.323, SIP or CONFCTRL)
- Number of participants – the number of participants for which GS maintains some information and can forward packets
- Participants' IP addresses – the address where the data can be forwarded to.
- Participants' port number - CCCS associates different types of applications with different well-known ports. For example, the H.323 stack uses port number 1720 whereas a SIP initiator will use port no 5060 for delivering control messages like invite a participant, request floor, leave etc.
- Current status (e.g. floor holder) – if a participant/port is sending data/audio that is not current floor holder, do not forward the packets to anybody else as a part of conference policy.
- Link status (e.g. broken link, slow link if possible etc.) – if a "Keep Alive" message didn't appear then delete the link

Client registration

A client must register itself with the CCCS before it can issue a request to the system or participate in a session. So for example, if a "CONNECT" call came from a reserved system port that is associated with a H.323 client, CCCS knows it will be a H.323 based client. The Gateway Services (GS) of the CCCS updates the registry with the port number of the conferee, the type (in this case H.323), the IP address, floor holder status (could be 0 or 1) and the link status (which is 1 if the link is alive). The GS continues to perform the operations of updating and adding the registry as participants come and go.

Session management

A client creates a session by specifying the initial attributes (passwords, policies for floor etc.). Creation of a new session involves two stages: negotiation of capabilities (like codecs) and allocation of resources. When a client wants to invite another client, the CCCS has to make sure that they both have at least an intersection of capabilities, for example, both are capable of understanding text or ascii values for data. The issues related to resource allocation is beyond the scope of this thesis. Session management also involves deploying a floor control policy. When a conference actually starts, MCS checks the floor holder status. If one of the ports is sending data that is not the current floor holder, GS does not forward the packets to all the other participants as a part of an implementation policy for floor control. Figure 20 is a representation of GS as a Broker system interoperating three different types of clients based on different architectures. This figure briefly illustrates the main operations discussed above.

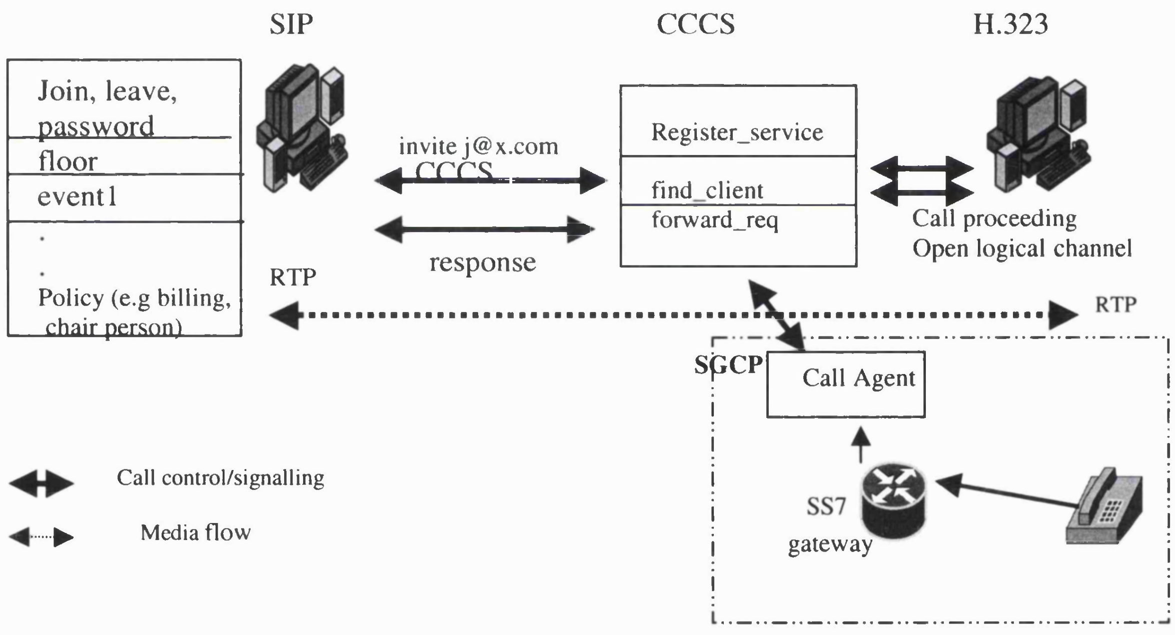


Figure 20: Topology of three different types architectures interoperating using CCCS

In the following section, the call setup and gatewaying different conferencing messages are discussed.

3.3.3 The CCCS's interoperability services

Out of the two features of internal management mentioned in section 3.3, the CCCS mainly provides interoperability, and this service is known as GS (Gateway Services). The job of providing the list of participants and their network information consistently is left up to Real-Time Transport Control Protocol (RTCP) up to a certain extent (RTCP provides a list of participants' geographic location in an RTP session in every few seconds. The interval gets longer if the number of participants get bigger). However, if GS receives a GS_LIST request then it sends out a list of participants that joined a conference via GS regardless whether they are capable of receiving RTP or not.

As an example of interoperability, the following section focuses mainly on the interoperability of IETF's SIP and ITU's H.323. As mentioned in chapter two, IETF and ITU's viewpoint on conferencing is almost opposite. Therefore, when designing a gateway that translates different functions between two completely different architectures, there are certain considerations have to be taken into account. The following sections look at these aspects of a conference control gateway.

3.3.4 Main conference control contrast between H.323 and SIP

To provide interoperability in conference control level between H.323 and SIP, the following issues are the most difficult to resolve.

- Modularity - The conferencing applications based on H.323 provide a complete package in one module. In other words, a H.323 based conferencing like Intel's Proshare will deliver audio, video and signaling facilities as a monolithic package.
  By contrast, as mentioned in section 2.3, Mbone based audio and video communication tools as well as a signalling protocols are independent protocols. They can operate as standalone packages. SIP is a session layer call control protocol for creating, modifying and terminating sessions with one or more participants. It is used to invite users and invitations used to create sessions carry session descriptions which allow participants to agree on a set of compatible media types. These similar

function are also provided in H.323, but they cannot be separated as a different module.

- Supporting protocols - In H.323 based systems, support for voice is mandatory, while data and video are optional. However, if data and video are supported, the ability to use a specified common mode of operation is required; so that all terminals in a conference can interwork. These modes of operation described in the ITU specifications are not necessarily provided in the Mbone based conferencing specifications. Recommendations in the H.323 series include H.225.0 packet and synchronization, H.245 control, H.261 and H.263 video codecs, G.711, G.722, G.728, G.729, and G.723 audio codecs, and the T.120 [T.120] series of multimedia communications. SIP mainly matches some of the functionality provided by H.225 and H.245 but also performs some other call control functions that are not supported in H.323.

- Advertising – H.323 does not use a standardised protocol to advertise its sessions whereas Mbone based conferencing uses Session directory Announcement Protocol (SAP) to advertise the sessions using IP multicast. NetMeeting uses Lightweight Directory Access Protocol (LDAP) to list the participants and their availability. Therefore, a publicly available seminar can be advertised over the Mbone will not be visible by a H.323 based application.

- Messaging – H.323 uses a binary representation for its messages, based on ASN.1 and the packet encoding rules (PER). ASN.1 generally requires special code-generators to parse. This makes it harder to debug the messages generated by H.323. SIP encodes its messages as text [Schulzrinne/Rosenberg NOSSDAV].

- Multicasting- H.323 supports UDP or multicast for user location, it does not currently provide for group invites. Therefore, although an IP multicast may be running as a transport protocol, H.323 cannot take the advantage from the network layer. SIP requests can be sent via multicast.

In conclusion, the main procedures to design a gateway like the GS are as follows: a) first of all, identify the functions that are incorporated in H.323 which are similar to SIP's main three requests (INVITE, ACK and BYE) . This will allow SIP users to at least join a common session. b) After that, follow the modes of operations that are described in the respective specifications. c) Identify if the user can use multicast capabilities. d) Finally, discard invalid messages and appropriate error messages should be sent different entities involved.

3.3.5 Basic Call set-up between H.323 and SIP

The following example shows interactions that takes place when a H.323 client invites a SIP client using the GS. Call signalling messages in H.323 may be passed in two ways. The first method is Gatekeeper Routed Call signaling (GRC). In this method, call signalling messages are routed through the gatekeeper between the endpoints. The second method is DiRect Call Signalling (DRC). In this method the call Signalling messages are passed directly.

In Figure 22, the calling endpoint Bell sends a set-up message to the well known port of callee (in this case, the GS on behalf of watson receives the calls to start with). The gateway then informs the caller that the call is being processed followed by the capabilities of the receiving terminal. It is not necessary that a terminal understands or stores all incoming capabilities; those that are not understood, or can not be used shall be ignored [H245]. Once the reliable H.225 control channel has been established, CCCS places the invite message in SIP format to the callee (watson) who is able to process SIP messages. Once the callee answers the call and both parties are prepared to interact, additional channels for audio, video, and data are established on the caller's side (based on the outcome of the capability exchange).

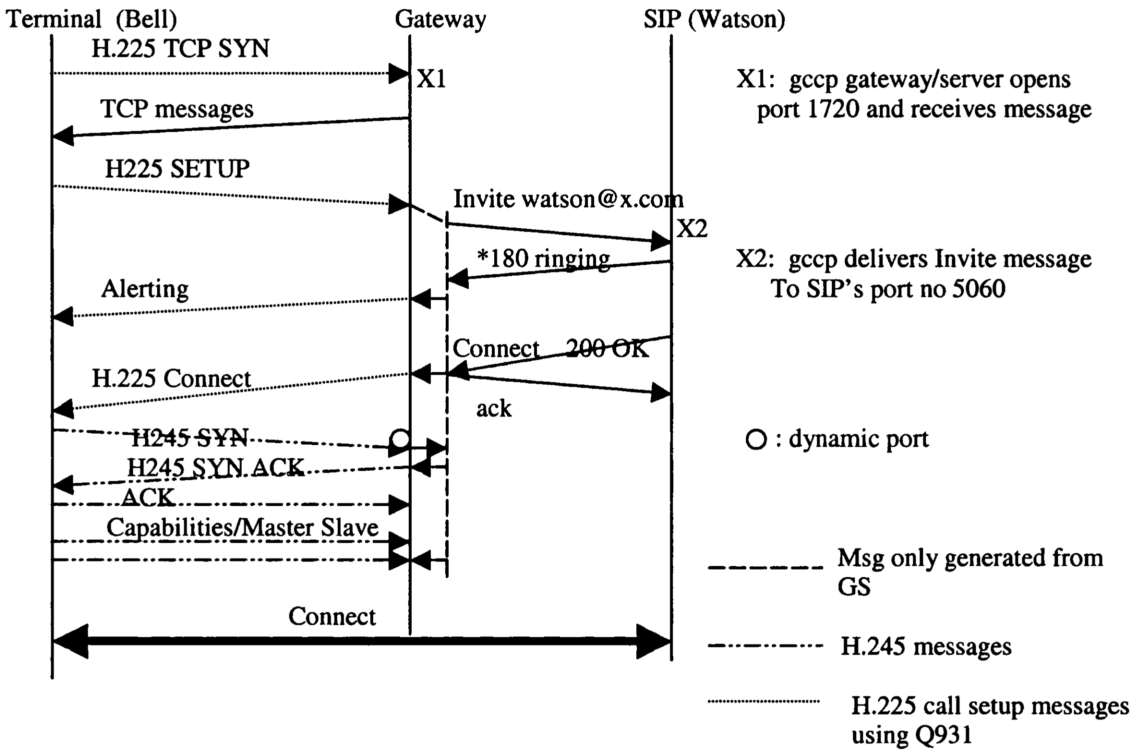


Figure 21: Messages exchanged between H.323 and SIP for "Invite"

* For two-party Internet phone calls, In the example above, Bell calls Watson which is
being translated by GS. A sample response to the invitation below, the 1st line is the status of client.
The Via headers are replaced as the req moves hop by hop towards invitee. Call-ID is unique in this invite.
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP csvax.cs.caltech.edu;branch=837 ;uaddr=128.16.16.16;ttl=16
From: Bell <sip:Bell@cs.ucl.ac.uk>
To: Watson <sip:Watson@x.com> ;tag=9883472 Call-ID: 296331305 Case: 1 INVITE
between the endpoints and this is the method we used in this experiment. Both methods use the same kind of connections for the same purposes, and the same messages. (see Appendix 2 for more explanation)

The response from GS to the caller is as follows:
SIP/2.0 200 OK
    Via: SIP/2.0/UDP csvax.cs.caltech.edu;branch=837 ;uaddr=128.16.16.16;ttl=16
    From:Bell <sip:Bell@cs.ucl.ac.uk>
    To: Watson<sip:watson@x.com> ;tag=37462311 Call-ID: 9883472
    CSeq: 1 INVITE

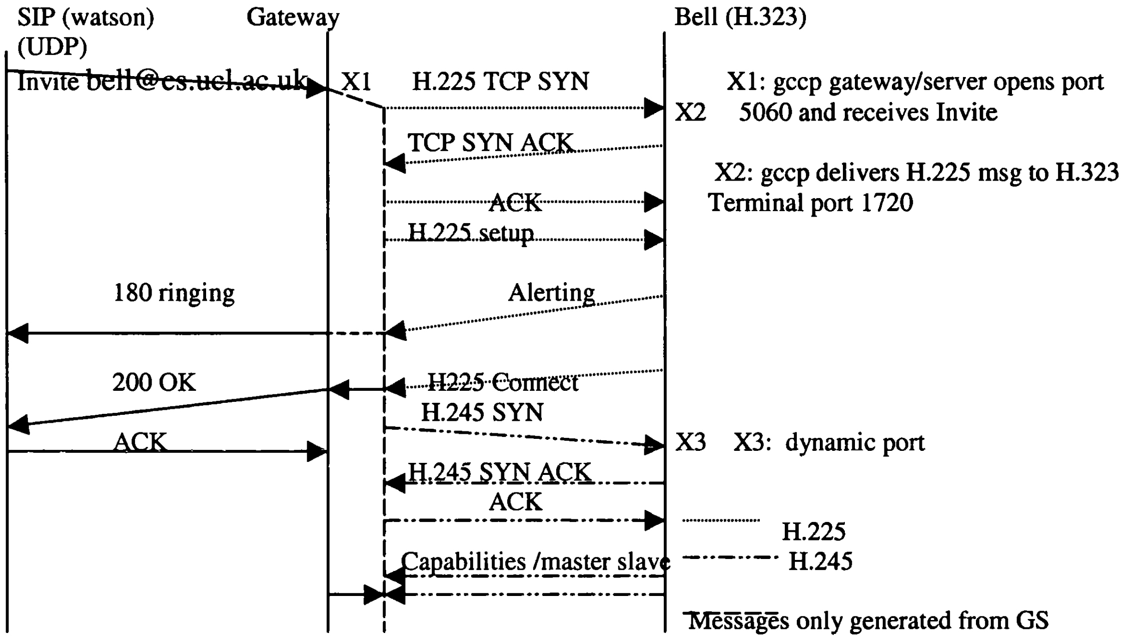


Figure 22: Messages exchanged between SIP and H.323

3.4 GS message format

The GS functionality of CCCS does not directly map the syntaxes of one protocol (e.g. SIP) to the other (e.g. H.323). It translates the syntaxes into semantics, i.e. it maps the conference control functions into CCCS's "native language", a text based protocol. This allows flexibility for future protocols to inter-work with each other. When an Invite message comes in to GS for example, it is translated to CCCS's native language, and then on the output it is mapped back to the necessary protocol. This gives CCCS a multi-protocol capability due to the fact CCCS maps most protocols' common functions to its common language.

The native language is based on text based protocol because it is easy to debug and monitor. Any request or response sent to the GS server or received from the server must contain certain fields in the following format in the following order:

Identifier	Value

"ID": *ConfID* – this is the conference ID of a particular conference (randomly generated ascii values) , 6 bytes long

"DestAddr": *Destination Address*- the IP address of where the messages should be delivered to in ascii text format

"SrcAddr": *Source Address* - the IP address of where the message came from in ascii text format

"ConfType": *ConferenceType* – the type of conference stack, normally upto 8 charecters long. Currently defined values are: H.323, MARS, CCCS, SIP, MBUS

"M": *Message* - It consists one of the following control messages:

CCCS_JOIN
CCCS_LEAVE
CCCS_INVITE
CCCS_LIST -request to see the participants in a conference
CCCS_FLOOR_REQ/ CCCS_FLOOR_ACC/CCCS_FLOOR_REJ

"R": *Reserved* – Left blank at the moment (for future purposes)

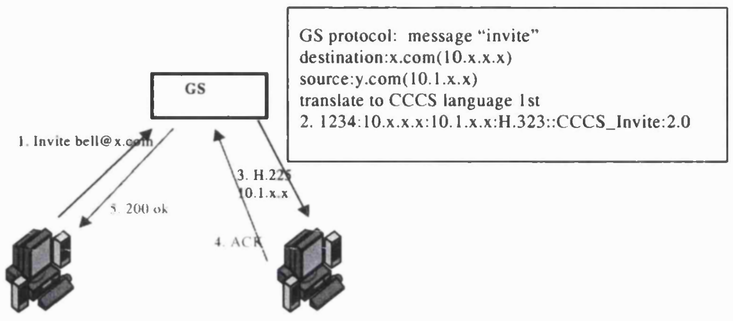
"V": *Version* – the version of GS server running

All the above fields are separated by reserved delimiters. A typical CCCS message for joining a conference would look like below:

8099111:128.16.8.88:192.8.9.0:H.323:CCCS_JOIN::2.0

ConfID:Destination IP:Source IP:Conference type:Message:Blank:Version of GS

Figure 23 depicts a basic example of three different protocols interworking with each other:



GS protocol: message "invite"
destination:x.com(10.x.x.x)
source:y.com(10.1.x.x)
translate to CCCS language 1st
2. 1234:10.x.x.x:10.1.x.x:H.323::CCCS_Invite:2.0

Figure 23: A simple example of interoperation between a SIP client, a H.323 client and GS client

1. A SIP client sends an "invite" message to a H.323 client (bell@x.com, where destination IP address is 10.x.x.x and source is y.com, 10.1.x.x)
2. GS "compatible" client receives the request and translates it to its own protocol (the message inside the box in the diagram) and checks x.com. The destination is a H.323 client
3. The GS client sends out an invite to H.323 client
4. H.323 sends an acknowledgement to GS client
5. SIP client receives acknowledgement.

Token objects by convention have upper case names, e.g., "FLOOR". Token can have zero or more holders(members). CCCS is responsible for maintaining (to a certain degree) a consistent list of all current participants and the applications that are in use.

3.5 Performance measures

Figure 24 shows **average** number of messages transferred between two compliant architectures like SIP and H.323. Table 6 shows the number of messages that are transferred between these terminals on different type of connections like over a LAN, a Dialup connection and WaveLan.

The average setup time between a SIP capable client and a Netmeeting client took on average 5 seconds using CCCS. Table 5 shows average results that were taken between H.323 capable terminals and a SIP terminal using CCCS.

CCCS

Time	No.of messages to setup connection	type of connection
3	50	LAN
7	55	WAN

Table 5: Number of messages transferred interconnecting SIP and H.323 using CCCS

SIP

Time[4]	No. Of messages	type of connection
2	6	LAN
2	10	Dialup
5	5	WaveLan

H.323

Time	No. Of messages	type of connection
5.50	30	LAN
4	25	Dialup
3	38	WaveLan

Table 6: Number of messages transferred and time taken between two H.323 terminal and two SIP terminals without a gateway

[4] Time (in seconds) taken to exchange control messages to set up the initiation

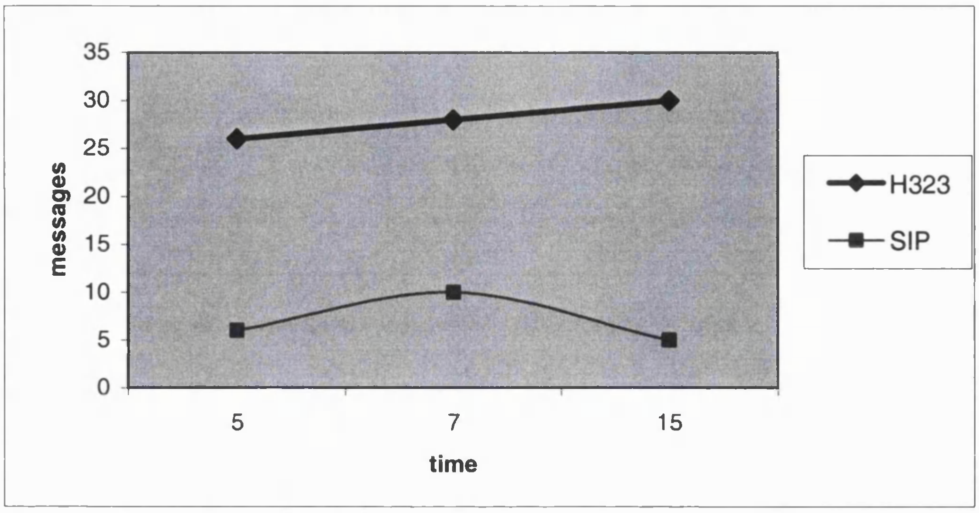


Figure 24: Average messages passed between H.323 and SIP terminals

In this scenario the SIP client was based in the USA and the Netmeeting i.e. H.323 client and the CCCS server were on the same LAN in UCL, London.

The tables show the number of messages that are to be expected. H.323 has a higher number of messages than SIP because H.323 has a sophisticated capability exchange mechanism by which the parties involved in the call can negotiate information like codec, speed and user information. Also, H.323 based applications are actually used to transfer audio, whereas SIP is used as a signalling protocol. However, we were interested in the actual number of messages that are exchanged between the caller and the callee. It is observed that, on average SIP has at least half the number of messages that are needed to be exchanged to set up the call. Both SIP and H.323 use TCP as the underlying transport protocol here. If we counted all the messages sent by H.323 after the setup has finished it is observed that on average in 8secs 85 UDP messages are sent (this is because the application is sending the silence packets).

The setup time between a H.323 terminal and SIP terminal which is running in the USA using CCCS is surprisingly low. The measurements were taken in the mornings in the UK time when the USA's network is less congested. However, the number of messages exchanged in order to perform the translation between two different architectures is quite high (shown in Table 5). That is because the GS is to translate simple JOIN, INVITE etc. messages in three different syntaxes. It translates for a CCCS compliant client as shown in Figure 20 and for a SIP and H.323 client.

3.6 Error control

The CCCS checks for system call failures and errors in user input. Fatal errors include failures in creating the socket systems calls on which to listen for TCP connection requests and failures to listen to multiple sockets due to blocking calls. If these occur, a message is written to standard error and the program exits. Nonfatal errors include errors in creating sockets for sending requests and reading or writing sockets. If a nonfatal error occurs, CCCS writes a message to standard error, sends an error code in its reply, or displays a message on the screen (depending on the error). For example, TCP sockets read errors include reading a request that is unrecognised or formatted such that it does not correspond with the formats shown in 3.4. If the socket is open, an error message is generated to the transmitter and the socket is closed.

Different conferencing stacks have different ways to handle failures. For example, normally, in H.323 the underlying reliable protocol of the H.245 Control Channel uses appropriate effort to deliver or receive data on the channel before reporting a protocol failure. Therefore, if a protocol failure is reported on the channel, the H.245 Control Channel, and all associated logical channels shall be closed. This will be done as if the endpoint had issued the H.245 endSessionCommand. With SIP there are mainly two places where error can be generated; either the server or the client. When a client generates a Status code 400 , it means the request contains bad syntax or cannot be executed at this server.

If CCCS receives any of the above error codes or messages it maps appropriate error messages to underlying protocol (e.g., it can send a 500 status code to SIP or send endSessionCommand for H.323).

3.7 Transport services

In a group communication scenario, any number of senders may be distributing information to the group or to a subset of the group. Each piece of information may be destined for any number of recipients. These messages can be either control messages or data. CCCS needs a way to distinguish the control messages from media data.

Any control messages (for example, CCCS_JOIN, INVITE , BYE etc.) will be sent to the GS server on the Unicast channel. For example, when H.323 sends one of its control messages, it uses the H.245 logical channel (see Figure 22-23). (If control messages arrive on a multicast port, the GS ignores those messages as control messages). However, if data arrives in multicast channel then the GS forwards those packets to all the possible recipients. If the clients are capable of receiving multicast messages, the GS multicasts it to them otherwise it uses the Unicast channel to forward data. This process allows the clients to be consistent.

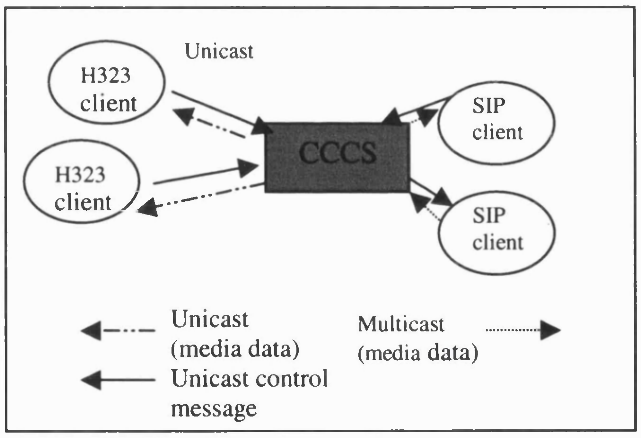


Figure 25: Methods of forwarding and receiving packets in CCCS

The simple diagram shown above scales only to a few number of clients per server. Beyond this, it is suggested that other peer servers be used to load balance the number of clients. However, the support for peering servers and issues related to this is beyond the scope of this thesis.

3.8 Intelligent Network operations

The future of CCCS entails in integrating some of the Intelligent Network (IN) features, like *authentication (AUTC)*, *call logging (LOG)* and *Call forwarding (CF)*. Therefore, it is useful to review some of the features that are already available in H.323 and SIP and highlight the features that are appropriate and needed to be implemented in either one of the architectures or in CCCS in future[5].

IN is an architecture for systems that provides services to enhance the basic call on a telephone network. ITU study group 11 published its accumulated descriptions of services and service features in annex B of ITU Q.1211,"Introduction to Intelligent Network capability set"[Q.1211]. Q.1211 divides the services it describes into two broad categories: "services", which are what an Intelligent Network vendor would actually wish to provide to customers; and "service features", which are lower-level building blocks used to construct the services. In section 3.8.1 the service features specified under CS-1 which are mainly appropriate for conferencing are listed. This section and Appendix C also provide a comparison list of whether SIP or H.323 are capable of providing these services and if it is already performs these tasks.

3.8.1 Service features

- *Authentication (AUTC)*

This allows verification that a user is allowed to access certain option in the telephone network. For the Internet, and for conferencing this is essential and both H.323 and SIP provide this feature. This service will need to be integrated in MCS part of CCCS.

[5] Please note details of all the other IN operations that are applicable to this thesis are appended in Appendix C.

- *Automatic call back (ACB)*

This feature allows the called party to automatically call back the calling party. At the moment, neither SIP or H.323 provides this. However, an email can be sent to the called party if they are not logged on and it is up to called party to call back.

- *Call forwarding (CF)*

This service feature allows the user to have his incoming calls addressed to another number, no matter what the called party status may be.

This is very easily provided in SIP using proxy server, in H.323 this also can be provided using H.450(3). Currently, CCCS does not provide this facility, instead if the callee is not logged on to the system, CCCS returns a message notifying the caller. Therefore, this feature will be advantageous to be implemented.

The services listed in Q.1211 which are not mentioned above or in Appendix C, include different billing options like credit card calling (CCC), freephone (FPH), premium rate (PRM) etc. However, these billing options are not really applicable for conferencing over the Internet. It is up to the end systems and users to negotiate payment methods and implement it accordingly.

There are some other new services that are introduced to latest draft of Q.1211. Call pick-up and message waiting are among those services that are provided by H.450(5) and H.450(7) respectively.

- Other services not in IN: SIP supports a number of features not supported in traditional IN networks. Among those is "OPTIONS" request that allows one end system to query another about what format it supports. Thus it provides capability negotiation. H.323 performs this function which is specified in ITU's H.245 recommendations.

SIP invitation requests have a number of optional parameters which traditional telephone networks lack, generally inspired by email. Addresses can have display name like From: x@hotmail.com. Messages can have Subject fields specified, giving a textual intended subject for a call. Users can have Organization fields, similarly giving the organization to which they belong.

3.9 Conclusion

In this chapter, a framework for communication services in conferencing has been presented. A major component in this framework is conference control which comprises of two major types of services: user visible and internal management. In this reference, an architecture for conference control has been proposed called CCCS (Common Conference Control Services). It provides a set of user visible services known as MCS (Main Control Services) and interoperates different types of architectures as a part of internal management features of a conference control. The latter part is referred as Gateway Services (GS) of the CCCS. Although significant effort is being put into defining how the existing telephone network services will interwork with the Internet in standard bodies, the main objectives of these proposals/projects are to define how voice-based services will work between these two networks. In this research, the CCCS is designed to provide a set of facilities that has the capability to provide conference control functions and the flexibility which is not present in one architecture on its own. Some of the major advantages include: a) H.323 based systems to do group invitations using Multicast b) users on the Internet to conference with an H.320 system running on the ISDN, therefore getting the network independence.

The Intelligent Network services like call transfer, answering services etc. are not yet implemented in this framework. In future, experiments on distributing GS server across the Internet are required to get performance measurements on how many conferees can interact with each other and how they can maintain consistency. In this chapter, the resource management and the network requirements for CCCS are not discussed. The next two chapters will be about network specifications for conferencing and Chapter 6 will be on resource management.

Chapter 4

Background of Reliable IP Multicast

This chapter and the next both deal with conference control framework from the network perspective. CCCS proposes to support and interwork two main models of conferencing over the Internet. As it was mentioned in the previous chapters, one of these models of conferencing (loosely coupled conferencing) works over IP multicast. IP multicast has some attractive features to provide for large number of participants in a conference and it can also support the CCCS framework. In this chapter we introduce IP multicast and review most of the available IP multicast protocols that can be used to support a generic model of conferencing. In the next chapter, we specifically point out the features that are required to support a conference control framework such as CCCS and those which need enhancements e.g. scalability and/or redundancy.

The main characteristic of time critical data used in desktop based conferencing (such as audio and video) is that they have very low delay tolerance and in the case of a loss, retransmitted traffic must not arrive out of order. Although a lot of research has been done to transport real-time traffic such as audio and video over the Internet, reliable delivery of conference control messages using IP multicast lacks a solution. One of the objectives of this research was to identify components that are missing for communication using different conference control protocols and reliable data delivery for conference control over the Internet is a suitable topic.

The Common Conference Control Services need support from its network to provide its services reliably. It also needs the network to provide:

- Effective retransmission in the case of loss
- Scalability
- Support for conference policies such as floor control and security

In this chapter, we review the network designs that meet the above criteria and analyse how it provides these mechanisms. Also we point out why IP multicast is suitable for large users based conferencing and appropriate for conference control model that claims to support that infrastructure.

4.1 Background of IP Multicast

As the Internet becomes more and more popular, the number of hosts that are connected to the Internet using multimedia conferencing increases. Therefore, the requirement to send data to many hosts simultaneously has been a major issue. In order to resolve the problem of providing data from many senders to many receivers in an efficient and in an inexpensive way multicast has been developed, with one of the first design and implementation put forth by Deering [Deering]. IP multicast effectively produces selective broadcast in which anyone can send a packet to a destination group address. The sending host is not aware of or participate in the complex route calculation; not need it take part in a complex signalling or call setup protocol. It simply addresses the packet to the right group, and sends it; certain nodes in the network replicate the data for all receivers. This is in contrast to Unicast replication (e.g., existing push technologies) wherein the source application replicates the data for each Unicast destination.

Adding multicast to the Internet does not alter the basic model. A sending host can simply send, but now there is a new form of address, the multicast or host group address. Unlike Unicast addresses, hosts can dynamically subscribe to multicast addresses and by doing so cause multicast traffic to be delivered to them [Crowcroft]. Thus the IP multicast service model can be summarised:

- Senders send to a multicast address

- Receivers express an interest in a multicast address

- Routers conspire to deliver traffic from the senders to receivers

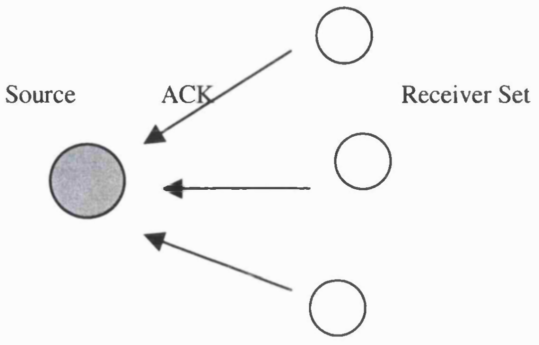
A multimedia application like video conferencing often involves a large number of participants and are interactive in nature with participants dynamically joining and leaving the applications. In order to provide many-to-many interaction when the number of participants is large IP multicast is undoubtedly a very good option for communication. In order to support a conferencing architecture from the transport or network layer one or more of these reliable IP multicast transport protocol may be required.

4.2 Approaches for reliable data delivery

When data has to reach several hosts over the Internet, there are issues concerned with reliable delivery. The primary goal for a reliable multicast protocol is to provide reliable delivery of packets to many destinations. Also dissemination on the scale of hundreds of participants scattered across the Internet requires carefully designed and flow and error control algorithms that avoid any potential bottlenecks. Generally the techniques for error recovery in reliable multicast fall in two categories: ACK based and NAK based, both of which employ a sequential numbering of data messages at the sender (as shown in Figure 26a). In an ACK based scheme, whenever the sender multicasts a data message, the receiver acknowledges its receipt by sending an acknowledgement (ACK) to the sender.

The other approach is a negative acknowledgement (NAK) scheme. Whenever receivers detect gaps in sequence number of message streams, they send repair requests (NAKs) to the sender as shown in Figure 26b. After receiving the NAK, the sender retransmits the data to the receivers. In general, none of the members keeps the group membership information. Since state information in this scheme is minimal, it scales well for a large data delivery model.

There are mainly two places where data can be retransmitted from: sender-initiated approach and receiver initiated approach. There is also a slight variation of receiver driven approach known as representative based retransmission.



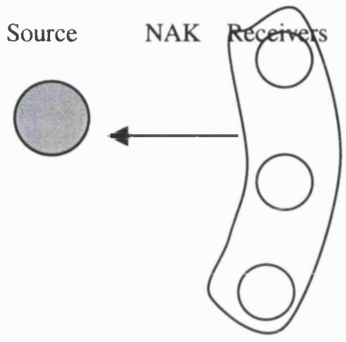


Figure 26a: A basic diagram of a sender initiated Protocol

Figure 26b: Receiver initiated Protocol

Earlier multicast protocols used conventional flow and error control mechanisms based on a sender-initiated approach in which the sender disseminates packets and uses either a Go-Back-N or a selective repeat theory mechanism for error recovery. If used for reliable dissemination of information to large number of receivers, this approach has several limitations. First, the sender must maintain and process a large amount of state of information associated with each receiver. Second, the approach can lead to a packet implosion problem where a large number of ACKs or NAKs are received and are processed by the sender over a short interval. Overall, this can lead to severe bottlenecks at a sender resulting in an overall decrease in throughput.

An alternate approach based on receiver-initiated methods shifts the burden of reliable delivery to the receivers. Each receiver maintains state information and explicitly requests retransmission of lost packets by sending negative acknowledgements (NAKs). Under this approach, the receiver uses two kinds of timers. The first timer is used to detect loss packets when no new data is received for a time. The second timer is used to delay transmission of NAKs in the hope that some other receiver might generate a NAK.

If another receiver in the group happens to have the data requested for, it sends the data to the requestor.

In representative based approach the receiver does not need to ask the source to resend the data, the receivers may ask its neighbour receivers or the ultimate parent node for retransmissions.

It has been shown that the receiver-initiated approach reduces the bottleneck at the sender and provides substantially better performance [Varadhan]. However, the receiver initiated approach has some major drawbacks. First, the sender does not receive positive confirmation of reception of data from all the receivers and, therefore, must continue to buffer data for long periods of time. The second important drawback is that the end-to-end delay in delivery can be arbitrarily large as error recovery solely depends on the timeouts at the receiver unless the sender periodically polls the receivers to detect errors. If the sender sends a train of packets and if the last few packets in the train are lost, receivers take a long time to recover causing unnecessary increases in end-to-end delay. Periodic polling of all receivers is not an efficient and practical solution in a wide area network. Third, the approach requires that a NAK must be multicast to all the receivers to allow suppression of NAKs at other receivers and, similarly, all the transmissions must be multicast to all the receivers. However, this can result in unnecessary propagation of multicast traffic over a large geographic area even if the packet losses and recovery problems are restricted to a distant but small geographic area[6][Sudan].

4.3 Design issues of reliable multicast

4.3.1 Organisation of nodes in a reliable multicast

The nodes in a reliable multicast are organised mainly in two ways: a) distributed model b) tree-based model. In a distributed model the nodes are distributed across the wide area

[6] Assume that only a distant portion of the Internet is congested resulting in packet loss in the area. One or more receivers in this region may multicast repeated NACKs that must be processed by all the receivers and the resulting retransmissions must also be forwarded to and processed by all the receivers

network. When a node requires retransmission of a message, it contacts one of its neighbours. The nodes keep a state table where it keeps a track of a limited number of closest neighbours which it can contact for retransmission as shown in Figure 27.

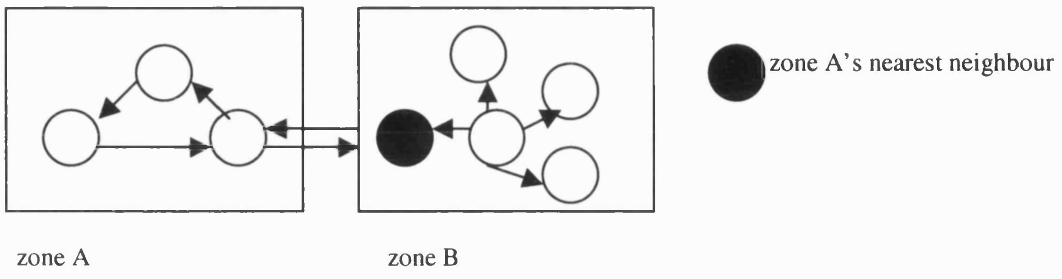


Figure 27: A distributed model of reliable multicast protocol

In a tree-based model the nodes are connected in a hierarchical structure as shown in Figure 28. Parent nodes which is referred as a Group Leader in Figure 28, keep track of its several child nodes. The parent node keeps track of other parent nodes which are closest to it. When a child node requires retransmission it contacts its local parent node. If the parent node has the data it retransmits to the node or it can contact other parent nodes to obtain the required data.

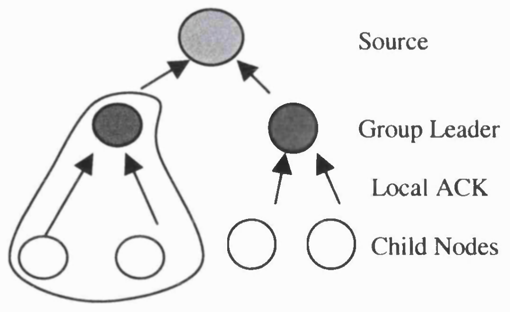


Figure 28: A basic diagram of a tree based reliable multicast protocol

4.3.2 Soft State vs Hard State

Several nodes involved in a reliable multicast transport protocol need to hold consistent information about their status. If a sender node fails or a network partitioning takes place, other nodes participating in that session need to be made aware of this. There are two main approaches to inform multiple nodes about their status: a) soft-state approach and b) hard-state approach. Soft-state approach is based around announce/listen protocol where the sender actively announces status updates and one or more receivers passively monitor and listen to those updates. This design principle works well in practice. Systems built on soft-state are robust. In [Clark D] Clark's "flow state" example, if a router crashes and the underlying path is recomputed, the flow state is automatically established along the new network path since periodic refresh messages from the end-system immediately begin to follow the new route.

In contrast, a hard state approach to flow state establishment would involve a specific setup and teardown protocol, e.g. Q.931 or ST-II. A benefit to this approach is that the state is established just once with a reliable delivery protocol like TCP, thereby avoiding the bandwidth or processing overhead of the soft state refresh messages. However, when a failure occurs in this environment, all the systems would have to simultaneously detect the failure and go through a complex procedure to explicitly tear down the old state and re-establish the new state along the new path. For example, if a router fails and routes are recomputed, all routers need to close down existing connection and then explicitly re-establish new paths. This is an exceptional condition that must be explicitly engineered and leads to complex interactions among many different distributed components [Raman].

In a soft state framework, consistency arises slowly, by virtue of the periodic announce/listen update process. As a consequence, the resulting design necessarily accommodates component failure and inconsistency intrinsically and thus can continue to operate in the face of adversity. For example, a multimedia conferencing system that relies on a centralised controller to track group membership or perform multiplexing would fail catastrophically if that controller goes down. In contrast, the loosely coupled conferencing model where distributed group membership is disseminated through announce/listen, gracefully accommodates end-system failures and network partitions. When a network partition occurs, for instance, the partitioned sub-sessions continue to operate and group membership knowledge that had spanned the partition eventually times out. Once the failure dissipates, the membership announcements resume their reach across the entire session and group state converges to track the reformed session. In summary, such designs are vertically robust in that they must accommodate inconsistency across distributed components throughout their design and as a consequence are robust against network pathologies so common to the Internet [Paxson].

4.4 Protocols

There have been few reliable multicast protocols that take into account the above design issues such as distributed vs tree based organisation of nodes, methods of retransmission and status updates. These features are required for different models of multimedia conferencing and conference control and it can be seen that most of the IP multicast protocols mainly used in the loosely coupled conferencing are based on a soft state based approach. Some of these protocols are described below:

4.4.1 Multicast Transport Protocol [MTP]

MTP [Armstrong] provides reliable, globally ordered and sequenced data between one or more processes. It is based on negative acknowledgements (NAKs). MTP distinguishes three different roles of members of a web (multicast transport group): master, producer and consumer. The *master* provides the message ordering synchronisation for all members in a web. The first member of a web becomes its master. *Producers* send data in messages (each sent as a sequence of data packets) after obtaining a token from the designated master. *Consumers* receive these messages and can use negative acknowledgements to request the retransmission of packets that did not arrive. This model can be suitable for a tightly coupled conferencing where the *master* functionality can support the MCU (discussed in Chapter 2) and the child nodes can be placed within the *consumer* channel.

The data transfer and retransmission is based on dividing time into heartbeat intervals (π). After the initial transmission of a packet, consumers have a limited time to request transmission of a data packet; this time is measured in heartbeats and is referred to as the retention (ρ). After that time, producers are no longer obliged to honour NAKs, allowing the producer to discard its copies of the data sent.

- Global ordering: We use the term "ordering" to mean that data is processed sequentially based on their Sequence ID. So, for example, audio and video data has to be "ordered", otherwise if they arrive out of order, the receiver wouldn't make any sense out of it. In MTP, the master is responsible for assigning global message sequence numbers to all messages. A producer that wants to send a message obtains a token by sending a Unicast packet (token[request]) to the master, which responds with a Unicast packet carrying a unique sequence number.

  Producers mark all packets of the message for which the token was granted with the message-number; they return the token implicitly with the final packet of the message (daa[eom]). Consumers are responsible for delivering messages in the correct order to their applications.

- Atomicity: A message may not reach a consumer correctly for two reasons: either the producer has failed and the consumer is failing/disconnected. In MTP, it is the responsibility of the master to provide atomicity. It maintains a message acceptance record, which assigns a status to the most recent 12 messages. If the master detects no data from the token holder for a while then it tries to confirm the status of the member. If the member does not respond, it is marked as reject.

  If a producer receives a NAK from a consumer requesting retransmission of 1 or more packets, these packets will be multicast to the whole group.

Although MTP is a carefully designed reliable multicast transport protocol and it works for certain applications (such as distant learning type of conference where there is only one lecturer), it has some drawbacks. The master function causes additional load on the machine on which it is being executed and its network connection, in particular global ordering and token processing. Every packet has to be processed by the master to update its view of the state of each producer. More significantly , the master is also a single point of failure.

4.4.2 Multicast Transport Protocol –2 [MTP-2/ MTP/SO]

MTP-2, later referred as MTP/SO [Ott J (b)], a variant of the original MTP, designed to avoid some of the original problems encountered and to provide some additional features. Like MTP, MTP/SO provides global ordering and it has three main groups of members: co-ordinator, repeaters and normal members. Messages from three different members are assigned to different streams, depending on priority. Therefore the delay caused by global ordering on a single stream primed by a single master is eliminated. MTP/SO proposes self-organisation of the members of a group into local regions for addressing the NAK implosion problem. MTP/SO provides a rate controlled transmission of user data. Additional features:

- MasterLoss: In MTP, the master is a single point of failure. Not only can the master fail, but a network partitioning also renders the partition without the master being inoperative. In order to achieve higher reliability in MTP/SO all members can detect the loss of the master when they do not receive any packets or new parameter values from the master. The recovery procedure starts by ascertaining that the master is indeed unreachable by sending it multicast master (suspected) packets. If no reply is received from the master, the members agree on a new master. A new web is formed using a new web-id, removing any ambiguity as to whether members still believe to be attached to the failing master. This design philosophy is based on the soft state based approach.

- Atomicity: MTP-2 handles atomicity in a more efficient way. In MTP, consumers only deliver a message to the application when they have received acknowledgement from the master that this message was "accepted" (this is indicated in the message acceptance record propagated throughout the web). It causes an increase of the transmission delay. MTP-2 defines a flag to disable the atomicity on a per message basis. Consumers are then free to hand messages to the application upon arrival of the complete packet sequence.

4.4.3 Scalable Reliable Multicast [SRM]

Scalable Reliable Multicast (SRM) [Floyd S], is a reliable multicast framework for light-weight sessions and application level framing. The SRM framework has been prototyped in wb, a distributed whiteboard application, which has been used on a global scale with sessions ranging from a few to more than 1000 participants. SRM is designed to meet only the minimal definition of reliable multicast, i.e. eventual delivery of all the data to all the group members, without enforcing any particular delivery order. The authors believed that, if the need arises, machinery to enforce a particular delivery order could be easily added on top of the reliable delivery service.

SRM attempts to follow the core design principles of TCP/IP. First, SRM requires only the basic IP delivery model – best-effort with possible duplication and reordering of packets – and builds reliability on an end-to-end basis. No change or special support is required from the underlying IP network. Second, in a fashion similar to TCP adaptively setting timers or congestion control windows, the algorithms in SRM dynamically adjust their control parameters based on the observed performance within a session. This allows applications using the SRM framework to adapt to a wide range of group size, topologies and link bandwidths while maintaining a robust performance.

- Framework: The framework is based on a distributed model where anyone can retransmit a lost packet. When receiver(s) detect missing data, they wait for a random time determined by their distance from the original source of data, then send a repair request. As with the original data, repair requests and retransmissions are always multicast to the whole group. Thus, although a number of hosts may all miss the same packet, a host close to the point of failure is likely to timeout first and multicast the request. Other hosts that are also missing the data hear that request and suppress their own request (this prevents a request implosion). Any host that has a copy the requested data can answer a request. It will set a repair timer to a random value that depends on its distance from the sender of the request message, and multicast the repair when the repair goes off. Other hosts that had the data and scheduled repairs will cancel their repair timers when they hear the multicast from the first host. (this prevents a response implosion). A lost packet ideally triggers only a single request from a host just downstream of the point of failure and a single repair from a host just upstream of the point of failure.

For a chain topology the essential feature of a loss recovery algorithm is that the timer value is a function of distance. For a star topology the essential feature of the loss recovery algorithm is the randomisation used to reduce implosion. Request/repair algorithms in a tree combine both the randomisation and the setting of timer as a function of distance.

With SRM's global loss recovery algorithm, even if a packet is dropped on a link to a single member, both the request and the repair are multicast to the entire group. In cases where the neighbourhood affected by the loss is small, the bandwidth costs of the loss recovery algorithm can be reduced if requests and repairs are multicast to limited area. One simple and now widely available mechanism for local recovery is the use of administrative scope in IP multicast. If a member believes that the loss neighbourhood and a potential source of repairs are contained in the local administrative neighbourhood, then both the request and the repair can be sent with administrative scoping. Also time-to-live (ttl) based scoping can be used to limit the reach of repair and request messages.

4.4.4 Reliable Multicast Transport Protocol [RMTP]

Reliable Multicast Transport Protocol (RMTP) is [Sanjoy] designed to send data reliably and effectively to large groups of simultaneous recipients. RMTP organises all the nodes into a tree structure. The receiving nodes are always at the bottom of the tree. Ideally, the sender is at the top, but this is not a requirement. The sender transmits messages using IP multicast. After a packet is transmitted, the sender will not release memory until it receives a positive acknowledgement from the group. The receivers do not send acknowledgements directly to the sender, but send hierarchical acknowledgements (HACKs). A receiver transmits a HACK to the their parent in the tree structure.

There are two different types of channels associated in RMTP's tree structure to avoid implosions and to send HACKs effectively. These channels are Data channel and Control channel. The data channel entities are divided into four classes:

- Sender (S) – Sender just sends data to the group.
- Receiver (R) – Receiver receives data and deliver data to applications.
- Designated receiver (DR) – A DR is a receiver that receives data, delivers data to applications, and buffers data for potential retransmission to its child nodes.
- Top node (TN) – A TN optionally receives unicast data and relays to the rest of the group. This option is designed for senders who do not support IP multicast.

Control channel entities are divided into five classes:

- Sender (S) – Sender just sends data to the group.
- Receiver (R) – It receives data and deliver data to applications.
- Aggregator (AG) – An AG accumulates ACKs and sends them to its parent node.
- Designated receiver (DR) – A DR is always an AG. A DR may handle local retransmission to its child nodes.
- Top Node (TN) – The TN is responsible for notifying senders of the global retransmission.

Finally, there are three types of retransmissions:

Global retransmission – Performed by the senders or the TN. The unacknowledged packets are multicast to all the receivers in the tree.

Sub-tree retransmission – Performed by the DRs. Usually the scope of the sub tree retransmission is defined by the TTL field in the IP header or is assisted by intelligent filtering in the multicast routers.

Local retransmission – Performed by DRs if the number of children missing the packets is less than a specific number (Lthresh). Local recovery is always sent through the control channel and is always unicast.

- HACK windows: RMTP solves the "ACK implosion" problem with a set of algorithms which strive to keep the control traffic received by any node in the tree, over any periods of time, in proportion with the data sent; by using HACK windows. A HACK window refers to a window of packets which a receiver or aggregator will respond to with a single HACK packet. The HACK window size W_{size} changes dynamically, and is never larger than H_{win}. Within a window, each receiver, DR, or AG is expected to send a HACK to its parent. To keep the load to the parent constant, RMTP uniformly distributes the time within a window at which each node sends its HACK.

4.4.5 Reliable Multicast data Distribution Protocol [RMDP] and Reliable Layered Congestion control [RLC]

In pursuit of the design of a one-to-many reliable bulk-data transfer protocol that runs on top of the Internet multicast service, there are two main issues have to be faced: reliability and congestion control. Vicisano et al [Vicisano97] handle reliability using Forward

Error Correction (FEC) techniques known as RMDP (Reliable Multicast data Distribution Protocol), and congestion control by means of a receiver driven scheme known as RLC.

- FEC and RMDP: RMDP is based around FEC, which operates on a principle where it anticipates some amount of losses, and resolves the loss by sending redundant data which allow the receiver to reconstruct up to a certain number of missing packets. The communication process thus includes an encoding phase at the sender, where redundant packets are constructed from the source data, and a decoding phase at the receiver, where source data are extracted, if possible, from the available packets.

  RMDP provides reliable delivery of data by Forward Error Correction (FEC) technique based on erasure codes [Rizzo97]. The basic principle behind the use of erasure codes is that the original source data, in the form of a sequence of **k** packets, along with additional **n** redundant packets, are transmitted by the sender, and the redundant data can be used to recover lost source data at the receivers. A receiver can reconstruct the original source data once it receives a sufficient number of (**k** out of **n**) packets. The main benefit of this approach is that different receivers can recover from different lost packets using the same redundant data. In principle, this idea can greatly reduce the number of retransmissions, as a single retransmission of redundant data can potentially benefit many receivers simultaneously.

  FEC can be computationally expensive, since the entire data stream must be processed by the encoder, so that each transmitted packet carries information on a (possible large) number of source data packets. Decoding can be expensive as well, depending on the encoding techniques being used and the actual of losses experienced. This renders the technique unattractive for unicast protocols. In reliable multicast protocols, though the advantages of FEC may overcome the encoding/decoding overheads[Rizzo97].

- RLC: Reliable Layered Congestion control (RLC) is the implementation of the receiver-driven congestion control. It is based on a redundant layered organisation of the data being transmitted. In this case redundancy is used not to provide robustness with the respect to transmission error, but flexibility with the respect of usefulness in receiving a given packet. For example, a receiver can obtain **n** data packets from **k** original source packets – so that what counts is the data received, not the actual information – i.e., dealing with packets, one can rebuild all the original **k** source packets once **k** packets are received out of the **n** sent, no matter which they are.

Given the redundant layered organisation of data, receivers are allowed to choose their receiving rate by means of joining the appropriate layers, this way the congestion control is performed independently from each other – although some co-ordination of nearby receivers might be needed to have more effective action on shared network bottlenecks.

The data and the redundancy packets are transmitted over *nc* layers, using a different multicast address for each layer. Receivers are allowed to choose their subscription level by joining/leaving multicast sessions. Receivers can control their receiving rate by choosing their subscription level. This way they establish a trade-off between bandwidth usage and receiving time. Here quality can be traded to save bandwidth. In order to allow receivers to fully exploit the bandwidth they use, the authors encode data using RMDP and organise it across layers.

On the sender side, packets of size **b** bytes are transmitted in all the layers. The receiver behaves as follows:

- If it is receiving a normal flow (non-burst) and sees a loss, drop a layer and setup a timer to prevent further losses, possibly due to the same congestion, causing another layer to be dropped before the timer is expired.
- If it has received the last packets of burst, and there have been no losses in this burst since it last joined the current layer (i), add a layer.

4.4.6 Pragmatic General Multicast [PGM]

Pragmatic General Multicast (PGM) is a reliable transport protocol for applications that require ordered, duplicate-free, multicast data delivery from multiple sources to multiple receivers [Speakman98]. PGM is specifically intended as a workable solution for multicast applications with basic reliability requirements. Its central design goal is simplicity and partial reliability for scalability reasons.

In the normal course of data transfer, a source multicasts sequenced data packets (ODATA), and receivers unicast selective NAKs for data packets detected to be missing from the expected sequence. Network elements forward NAKs PGM-hop-by-PGM-hop to the source, and confirm each hop by multicasting a NAK confirmation (NCF) in response on the interface on which the NAK was received. Retransmissions (RDATA) may be provided either by the source itself or by a Designated Local Retransmitter (DLR) in response to a NAK, or by another receiver in response to an NCF. Since NAKs provide the sole mechanism for reliability, PGM is particularly sensitive to their loss. To minimise NAK loss, PGM defines a network-layer hop-by-hop procedure for reliable NAK forwarding.

Upon detection of a missing data packet, a receiver repeatedly unicasts a NAK to the last-hop PGM network element on the distribution tree from the source as shown in Figure 29. A receiver repeats this NAK until it receives a NAK confirmation (NCF) multicast to the group from that PGM network element. Finally when the source itself receives the NAK, it confirms by multicasting an NCF to the group.

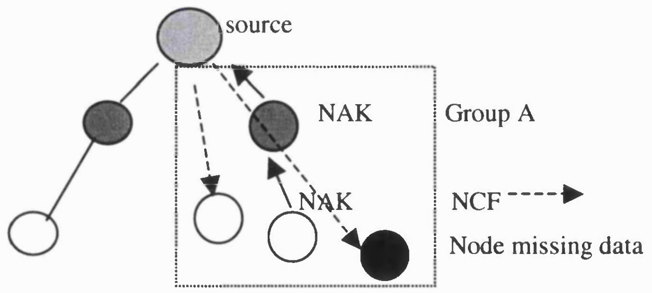


Figure 29: A basic diagram for PGM

Besides procedures for other receivers to provide retransmissions, PGM also specifies options and procedures that permit designated local retransmitters (DLRs) to announce their availability and to redirect retransmission requests (NAKs) to themselves rather than to the original source. In addition to these conventional procedures for loss recovery through selective ARQ, Forward Error Correction (FEC) can be used for sources to provide and receivers to request general error correcting parity packets rather than selective retransmissions.

As a further efficiency, PGM specifies procedures for the constraint of retransmissions by network elements so that they reach only those group members that missed the original transmission. As NAKs traverse the reverse of the ODATA path (upward), they establish retransmit state in the network element which is used in turn to constrain the (downward) forwarding of the corresponding RDATA.

4.4.7 Example of reliable multicast embedded application: Network Text Editor [NTE]

Network Text Editor (NTE) [Handley 97] is a shared text editor which runs on top of IP multicast. While this is an application like wb, it has a reliable multicast transport protocol embedded in it. The data distribution model uses the redundancy achieved through treating a line as an ADU (Application Data Unit) combined with the fact that most successive modifications are to the same line to avoid the need for most retransmissions.

The reliability of the underlying IP multicast transport protocol is based on a distributed, replicated data model, where every participant holds a copy of the entire document being shared. End-systems or links can fail, but the remaining sites still have enough data to continue if desired. Because of the nature of the application, it is important that all the recipients have consistent data on their screen, however inconsistencies can result from packet losses, effectively simultaneous changes to the same object. However, the authors proposed a mechanism that ensures inconsistencies are resolved, irrespective of the number of packets lost.

There are three parts to the inconsistency discovery scheme. Two mechanisms are based on session messages being sent out periodically by each site[7]. To detect inconsistencies, each session message carries a timestamp and a checksum of all the data. If the timestamp given by another site is later than the latest change a receiver has seen, the receiver can request all changes from the missing interval without knowing what the data actually was. This may not fill in sufficient information to ensure consistency, so checksum is used to discover that a problem has occurred. This is followed by an exchange of checksums to discover which blocks the differences are in.

The third mechanism is designed to prevent the above mechanisms from needing to be used where possible. There is a concept of current site (this is the site which has most recently been active) which multicasts out a summary packet giving the timestamps and IDs of all the most recently changed objects. If a receiver has a different version of one of these objects then it is entitled to either request the newer version from the current site, or to send its newer version.

- Scalable retransmissions: When a receiver discovers there is an inconsistency between its data and that of another site, it cannot just send a message to resolve the inconsistency immediately because it may cause NACK implosion. SRM uses the mechanism of retransmitting requests that are delayed by a random period of time partially dependant on the round-trip time between the receiver and the original source. Requests are then multicast and serve to suppress further duplicate requests from other receivers. As it has no redundancy mechanism, wb's SRM implementation is more dependant on its retransmission mechanism than NTE is, and thus it requires its retransmission scheme to be extremely timely. NTE does not wish its retransmission scheme to be so timely, as it expects most of its loss to be repaired by the next few characters typed. This results in very significantly fewer packet exchanges because in a large conference on the current Mbone, the probability of at least one receiver losing particular packet can be very high. Thus what is required is

[7] these messages are sent out at a rate that is dependant on the total number of sites in a conference to keep the message rate low.

a retransmission scheme that ensures that genuine inconsistencies are resolved in a bounded length of time, but that temporary inconsistencies due to loss which will be repaired anyway do not often trigger the retransmission scheme.

4.5 Conclusion

In this chapter, IP multicast has been introduced which can be used to support large scale conferencing and the CCCS framework from network perspective. Reliable and ordered data delivery are two essential services required from real-time applications such as conferencing. In this chapter, different approaches for reliable data delivery and ordering have been presented. There are mainly three structures of reliable data delivery in IP multicast: centralised, distributed and hierarchical. Reliable IP multicast protocols like MTP follows a centralised delivery system, SRM, NTE fall in the second category and RMTP and PGM fall in the hierarchical category. SRM, RMTP and FEC provide most of the important features that are required for reliable IP multicast. Although this chapter provided some design criteria for reliable multicast and conferencing, it is in the next chapter, where we identify the main features of these transport protocols for transporting conference control.

Chapter 5

Reliable IP Multicast Transport Protocol Requirements for Conference Control

A multimedia application like video conferencing often involves a large number of participants and is interactive in nature with participants dynamically joining and leaving the application. In order to provide many-to-many interaction when the number of participants is large IP multicast is a very good option for communication of control and data. IP multicast provides scalability and efficient routing but does not provide the reliability these multimedia applications may require. Though a lot of research has been done on reliable multicast transport protocols, it really seems that the best way of doing a reliable multicast is to build it for a given purpose like conference control in multimedia conferencing.

This chapter compares some of the reliable multicast transport protocols described in Chapter 4 and analyses the most suitable features and functionality provided by these protocols for a facet of conference control, floor control. The goal is to find or design a reliable multicast transport protocol which would suit the CCCS framework, i.e. scale to a large number of participants scattered across the Internet and deliver conference control messages reliably.

5.1 Problem scenario

The Mbone based conferencing protocols and applications have always been designed to work over IP multicast, whereas some features such as the call control functions of tightly coupled conferences like ITU's H.xx series of recommendations have only recently been designed to work with TCP and use UDP for data and audio. Applications that run over Mbone are designed to cope with the unreliability of the underlying network and normally work in a distributed manner over basic IP multicast. In contrast, H.xx series based applications deploy a centralised model which normally run over reliable unicast links. A generic model for transport protocol which would cater for both these models of conferencing needs to provide one of the following:

a) a function that can connect a centre to a decentralised system
b) a function that adds state to the connectionless system such as the Mbone based protocols
c) a function that can provide reliable multicast and connect it to reliable unicast

This chapter mainly concentrates on option c) where existing conference control protocols based on different architectures can be supported by the network.

As discussed in Chapter 4, IP Multicast provides a service model by which a group of senders and receivers can exchange data without the senders needing to know who the receivers are*, or the receivers needing to know in advance who the senders are. Hosts that have joined a multicast group will receive packets sent to that group. Therefore, this service model can lead to applications which will scale to hundreds/thousands or more receivers. However, because of the limited bandwidth most applications like videoconferencing will often deploy floor control to limit traffic from the group to a small number of concurrent sources.

In order to support floor control either for a tightly coupled session (where reliability and ordering of the messages may get the highest priorities) or a loosely coupled session (where congestion control or retransmission strategy may be more complex and more critical than strict ordering), certain characteristics from a multicast protocol are required. The requirements for conference control from a transport protocol are:

1. Loss detection and successful reliable delivery
2. Retransmission strategy, queue management
3. Scalability - source to many receivers, many sources to many receivers etc
   Ordering
4. Scope of membership

* Unless a higher level agreement has been done.

5. Congestion control
6. Integrated security

A lot of research is being done on reliable multicast transport protocols. This chapter highlights some features of most popular and standardised multicast transport protocols around at the moment and compares them against the requirements of single facet of conference control, Floor Control. Most of the reliable multicast transport protocols are discussed in details in chapter 4.

5.2 Key Design Issues in IP multicast Transport Protocols

Loss detection and retransmission strategy are two important aspects in the design of any reliable protocol. In a reliable transport protocol a recipient can (within bounded time) find out when it is failing or being partitioned from active senders. A sender is assured (with sufficient probability) that all its messages reach within bounded time.

In a traditional point-to-point reliable protocol such as TCP, positive acknowledgements are used to detect loss and the sender is responsible for retransmission of the packet. Using TCP one can provide HTTP Web traffic, FTP file transfers, and e-mail. All TCP traffic is unicast, that is it has one source and one destination. The nature of data can be either bulk data transfer where all data is sent one way and then the sender waits for a response or interactive where as soon as each data unit is sent acknowledgement has to be returned. The transmitter sends out a window's worth of data before requiring an acknowledgement.

It is harder to transfer data "reliably" from source(s) to **R** receivers (where R can be 10's to 100,000 or more), because multicast protocols interact with multiple parties simultaneously and so involve a higher number of links. Therefore, the likelihood is greater that some of the paths in the source's multicast tree are unstable at any time. In addition, the instability in any portion of the multicast tree may affect many members of the group because of the collaborative adaptive algorithms used [Floyd]. In particular, it is difficult to build a generic reliable transport protocol for multicast, much as TCP is a generic transport protocol for unicast. Reliable multicast is a case where "one size fits all" does not work at all. Applications often have very different reliability and latency requirements, state management styles, error recovery and group management mechanisms. A reliable multicast transport protocol that meets the worst-case requirements is unlikely to be efficient and scalable for many application requirements [Zheng].

In a teleconferencing environment, a desirable robustness property is the ability to continue operating within partitions should the group become fragmented. Ultimately, the applications that use the multicast transport platform should be the ones to decide when the situation has deteriorated to a point where continuation is meaningless.

5.3 Floor control and its requirements

A floor is an individual temporary access or manipulation permission for a specific shared resource, e.g., a telepointer or voice-channel, allowing for concurrent and conflict-free resource access by several conferees. For example, a floor requester in a meeting room would be a person who raises his/her hand up to ask a question. In this situation, it is up to the chair to grant the floor to the requester, although other mechanisms for assigning the floor exist. The session parameter includes the number of collaborators, and their role (chair, listener, a floor holder), determining their capabilities.

There are different schemes of floor control as discussed in section 3.2. Whatever the scheme is, for applications to scale beyond a few participants, all communication must be multicast. Some research has been carried out to support Interactive collaboration application like TMTP [Sudan] for data, STORM [X.Rex] for audio and video and SRM for wb. However, the nature of floor control is different to these interactive applications. For example, the volume of data i.e. floor control messages are lot less than audio or video or whiteboard associated data, the timing of requesting/granting floor control can be very specific (for example, when the chair/speaker addresses the audience and asks for questions, a lot of listeners are going to request the floor but before that traffic may be lot less), ordering of data is more crucial factor than audio/video (for fairness, or applications like when customers are bidding for share) etc.

Typically traffic control for floor requests would be done in low level per source. An example of sudden flood of traffic would be "Flash Call" problem in POTS. Flash call would occur when a televoting system is taking place, where the viewers call a telephone number provided by a particular program, to give their opinion. The first method to avoid this sort of problem is the nondeterministic approach, where after certain calls being taken by the network, users would hear an equipment engaged tone. This would stop the network being flooded by too many calls. Other approach is the deterministic approach, where the telephone company would be warned in a day advance, by the programme organisers. So the telephone company can provide enough resources for that sort of service, and the cost would be higher.

On a data network, a similar situation can also occur. There are certain traffic problems which only apply to floor control and conference control type of applications. A reliable multicast protocol has to include certain features which would account for:

- *Congestion control* - The volume of traffic will increase at certain times. The reliable multicast has to cope with sudden burst of traffic. Many sessions have precise starting times, causing a lot of message generation when most of the members of a conference join the session.
- *Ordering* - To be fair to all the floor requesters the protocol has to have a mechanism for strict ordering. It has to be consistent and fair so on average everyone is treated fairly.
- *Reliability* - To provide good services, reliability and the retransmission strategy is very important. Assume the scenario, where a floor request is multicast by A, B didn't receive the message after time **t**. B now bids for the floor, without knowing the floor requester is A. Imagine there is a policy in this conference that if someone has requested a floor, the next person is not allowed to bid for the floor within next **t'** seconds. Now somehow in this scenario, someone has to inform B that A has asked

for the floor, and he may not request/being granted the floor. There are several ways to resolve retransmission which depends on protocols design, for example, in TMTP the domain manager retransmit the data, whereas in SRM the nearest receiver to B will transmit the data.

- *Member Classes* - There can be different types of members in a conference. The rate controlled transmission of user data is very useful for floor control. For limited bandwidth, this is a way to limit number of concurrent users on the network. For example, one type of member will be not just a member but also a potential co-ordinator and repeater. Another type of members will be just normal members, the last type of member will unreliable receiver who will not ask for retransmission. If the members are categorised like that then the job of the application programmer is made a lot easier. A model like MTP/SO proposes to meet this requirement.

5.3.1 Functional Criteria

The table below is a comparison of several multicast transport protocols based on functions that are relevant for floor control. It is a summary of the protocols' retransmission scheme, floor control requirements, delivery unit which highlights the cost of deploying these protocols on the network:

Table 7: Comparison of multicast transport protocols for floor control

Protocol	*Reliability Semantics*	*Congestion Control*	*Participant structure*	*Knowledge of participant*	*ACK / NACKs/ Retrans Mission*	*Unit of delivery*
SRM	Reliable	No	Distributed	Via session messages	NAK, receiver reliable	1 ADU = app. Data unit
RMTP (BELL Labs)	Reliable	Yes	Hierarchy of regions, Domain regions	Optional, May be known	Window of packets ACK/ HACK	N = window size

RLC + RMDP	Reliable	Yes	No	No	No ACK/ NAKs FEC for error recovery	K/N = depend on file size
PGM	Reliable	No	Local retrans-mitters	No	Bread crumb	1 packet
MTP/SO	Reliable, totally ordered, atomic delivery	Through different streams	Master Repeater Consumer	Known	NAK (?)	?
NTE	Reliable	No	Distributed	Via Session packets	Triggered NAKs with randomi-sation + FEC	1 ADU = 1 packet

5.4 Analysis of available reliable IP multicast – limitations, advantages and disadvantages

SRM: One of the problems with SRM is that this algorithm will end up consuming a lot of bandwidth when there is little correlation of losses among receivers. For example, in a group of 1000 receivers, when only one receiver loses a packet, all 1000 receivers need to process the multicast NACK and repair packets. This causes significant overhead for the hosts. Also if one set of hosts in particular requires a packet, it is not desirable to multicast the packet to all the possible groups. One possible method of improving SRM's efficiency is to use localised recovery. The idea is to multicast NACKs and repairs locally to a limited area instead of to the whole group. Using the TTL (Time to Live) field in the IP packet header is one possible way to implement scope control. However, SRM does not deal with congestion control that may be caused by flood of packets when a session starts or question time for a conference for example.

MTP/SO or MTP-2: MTP-2 has been designed to work as a multicast extension for tightly coupled conferences like T.120 standards. The specification of T.120 standard emphasises that all T.120 nodes need to be backward compatible, i.e. if a node is not capable of running over IP multicast then there must be a way to build a connection between a multicast capable and non-multicast capable T.120 node. Therefore, another protocol has to accompany MTP-2, known as MMAP (Multicast Adaptation Protocol). When a connection is built between two T.120 MCS provider, MMAP is firstly used to setup the connection and then determine if these nodes are capable of running MTP-2 as a reliable multicast. This causes a lot of information and data exchanges before the nodes can be joined together. Also, it is built around the master that performs the required co-ordination functions: rate control, global ordering, handling join and leave requests. This has the drawback of Single Point of Failure. However, The rate controlled transmission of user data in MTP-2 is very useful for floor control. If only few users are capable of holding the floor then there is only little point of giving all the other 10,000 receivers the capability of asking for retransmission of floor request.

RLC/RMDP: RLC/RMDP are very good mechanisms for bulk data transfer such as ftp. They do not really satisfy the needs for floor control. For example, in floor control mechanism the identity of the participants are quite crucial. Combination of RLC and RMDP is not really appropriate for floor control purposes due to lack of timely delivery to a single source.

PGM: PGM, only a semi-reliable protocol, is not intended for use with applications that depend either upon acknowledged delivery to a known group of recipients, or upon total ordering amongst multiple sources. For floor control, these two functionality are quite crucial, therefore PGM is not the best suited protocol for floor control. PGM is better suited for applications in which members may join and leave at any time, and that are either insensitive to unrecoverable data packet loss or are prepared to resort to application recovery in the event.

RMTP: RMTP seems to be the best protocol suited for conference control. MTP's control channel and data channel assures different level of group membership and reliability accordingly. Also different members like Designated receiver (DR) and Top

Node (TN) can assure retransmission while reducing the risk of congested links. The HACK window also is a good mechanism for buffer management. In section 5.6, Gossip-style Garbage collection method is discussed which could be used in RMTP for managing retransmission buffers.

Summary: Many protocols are proposed and implemented:

- Protocols differ widely in design
- Logical structure of communication pathways (ring versus tree versus none)
- Group membership mechanisms and assumptions
- Receiver-reliable versus sender reliable
- ACK/NAK and FEC

Based on floor control requirements from a reliable IP multicast (as discussed in section 5.3) RMTP will be one of the most suitable transport protocols because of the reasons stated above. Also SRM will be a suitable protocol for this purpose because it represents a simple and robust approach for large-scale recovery based on persistent state, suppression of duplicate NACKs and repairs, and global retransmissions. The messages specify a time-stamp used by the receivers to estimate the delay from the source, which causes global ordering. However, if the number of participants is very large, the convergence time will grow exponentially and SRM will not be the best suited algorithm.

If some of the participants in a video conference is unicast only a tree based structure for IP multicast like RMTP or MTP/SO will be quite suitable. In the hierarchical system, one parent node can have several unicast only child nodes underneath it and it can unicast the data to these child nodes. In this model the participants list can be viewed by the parent node.

5.5 Limitations of floor control

A lot of the multicast transport protocols like SRM, RMTP, MTP/SO will meet some of the requirements for floor control. Certain protocols can be customised or adopted to meet some of the requirements. However, there are some limitations of a floor control mechanism itself because of the nature of its behaviour. The principal difficulty is in achieving scalability to large group sizes. In a conference, where all members have access to the ability to request (and grant) the floor, it is necessary for all participants to know approximately who the other participants are. Also it is essential to see who are bidding for the floor, otherwise, none can see a global reason for giving someone the floor.

If the access bandwidth is small compared to network backbone bandwidth, at time t, there may be 1000 receivers in the system, however using RTCP the report of the participants may show only first 20 participants*. To account for congestion control a solution has been suggested in timer reconsideration for enhanced RTP scalability [Rosenberg]. In a multimedia session which is using RTP/RTCP for transporting audio and video where RTCP rate is 1 kb/s. If all RTCP packets are 1 kb, packets should be sent at a total rate of one per second. Under steady state conditions, if there are 100 group members, each member will send a packet once every 100 seconds. However, if 100 group members all join the session at about the same time, each thinks they are initially the only group member and sends a packet at a rate of 1 per second, causing a flood of 100 packets per second or 100 kb/s, into the group.

So the effect of timer reconsideration algorithm is to reduce the initial flood of packets, which occur when a number of users simultaneously join the group. A participant P who wants to join at time t will determine the group size and it will transmit at time t', where t'> t. So if a session has to start at 10:00 am, packets will be sent at 10:01 am, 10:02 am and so on. Therefore, at time t, the report showing the number of participants at 10:00 am will not be correct.

So the underlying technology has to support users to join a session at t" where t" < t. In other words, if the session is programmed to be broadcast at 10:00 am, users have to join

* If the reliable protocol is distributed (e.g. in SRM/NTE)i.e. the participants can only see the local information straight away and overall statistics is an option, then this problem can be eliminated to an extent.

the session from 9:55 am. That requires modification of connection charges to include the traffic flow pre session.

If each participant sends messages at the rate of K/N per second, where K is the fraction of total capacity allowed for the RTCP messages, the following can be derived:

For audio, we might choose to have 1 speaker and therefore K is the capacity of that 1 flow. Typically RTCP messages might be limited to 5% of the flow, so for 20 packets per second, we would be allowed 1 message per second. Over 5 minutes, this would allow N to reach 300.

For video, we may choose to allow either one video to flow to several participants. To save bandwidth, we probably choose the current speaker's video channel, which might be sending 100 packets per second from each and every source, which allows for K=5, or N to reach 1500 participants after 5 minutes.

5.6 Proposed solution

The goal of this reliable IP multicast is to maximise the performance of the resident applications like multimedia conferencing. After discussing the advantages and disadvantages of the different protocols it seems that a reliable multicast protocol has to be able to provide:

Congestion control: The protocol has to cope with sudden burst of traffic. A mechanism has to be provided where pre session traffic flow is allowed. RTP timer reconsideration [Rosenberg] is an example to deal with congestion control. Also if a user who just got the floor waits a certain amount of time before asking for the floor again will help the implosion as well.
Ordering: The point about floor control is that requestors should get a fair chance at getting the floor. The problem with the reliable multicast transport protocols is that to scale, they use techniques like SRM (random timer). What is required is a deterministic

(round robin) timers for people requesting the floor at the same time. So if a participant asked for the floor or got the floor last time, then they have to go after everyone else - i.e. that user/participant has to wait before asking for the floor again.

Reliability : The protocol has to retransmit lost/damaged packets reliably. Not just the source, any one holding the packet will transmit the packet to the receiver require that damaged packet. SRM's retransmission strategy provides that.

Distributed control: There is a limit on the size of conference of known participants because convergence time increases as the number of users increase. A hierarchical system with just the knowledge of certain group or certain local users will be a possible solution. RMTP or STORM can provide that sort of architecture.

Simple: A protocol should be easy to implement and enhance.

Other: Able to cope with unicast only receivers. A security mechanism will be added advantage.

Proposed solution 1:

A proposed solution for distributing conference control over reliable transport protocol consists the above features, where the group members/nodes are formed into a hierarchy. In the section below we provide an overview of a hierarchical protocol that has already been deployed on application layer known as HGCP [Dommel]. The similar design principle of this architecture can be applied on the network layer to provide a reliable transport protocol suitable for conference control and floor control.

In HGCP, the nodes are assigned tags as shown in Figure 30 where the top node is labelled as **1**. In order to reduce the effect of a sudden burst of traffic, nodes no 111 and 110 for example, will not send their requests directly to node no. **1** but will request for the floor to their local parent node (in this case 11). Please note that, the following does not give a detailed specification for a reliable multicast protocol. It only highlights how a protocol can be designed to cope with congestion control, reliability and ordering and yet simple to implement.

The source station is the current floor holder and transmits information to the receiver set; hop nodes are positioned on the path from the source to receivers. The propagation tree of HGCP organises group participants into a hierarchy of subgroups or *coteries*. Each such coterie has a group manager, which acts as a representative for all other members in a group. The group manager is responsible for querying control states for its group members. This group manager can also be responsible for RTP reconsideration to control sudden burst of traffic.

The HGCP protocol consists of two stages: 1) Propagation tree construction 2) Control message dissemination.

Figure 30 shows a sample HGCP scenario. In order to implement a protocol like HGCP in the transport layer the nodes will be labelled in transport layer instead of in application layer where the parent nodes like 11 or 10 can deploy RTP timer reconsideration to control the sudden burst of traffic.

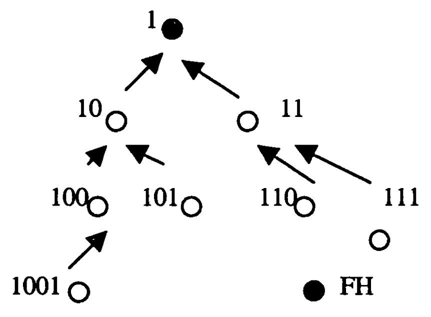


Figure 30: Tree structure for hierarchical transport protocol

Relating back to reliable multicast protocols discussed in Chapter 4 and section 5.4, it can be envisaged that RMTP follows a very similar pattern to that of HGCP. The Top Node (TN) and the aggregator functionality can be labelled as 10 or 11 in Figure 30, and the child nodes (1001 or 110 and alike) can be either senders or receivers.

Proposed solution 2:

Apart from the method described above which meets conference control requirements, some of the existing protocols can be adapted further. For example, hierarchical ACK based protocols like RMTP and NAK based protocols like SRM can combine a method as described below:

Gossip style – Gossip Style Garbage Collection (GSGC) [Guo] is a stability detection framework intended for reliable multicast at the transport level using session messages. At minimum cost, the GSGC service offers failure detection, stability detection and buffer management to existing large scale reliable multicast protocols such as RMTP and SRM. GSGC design principle can be applied to an existing protocol mainly to keep group membership information stable and detect failures in a session, however the GSGC topology does not deal with congestion control or atomicity.

The stability detection algorithm works as follows: there are **m** senders in a group of size **n** and each sender uses an independent sequence space. Each member maintains an array **R** with **m** number of slots in it. The *j*-th element of this array **R** is the maximum sequence number such that all messages with less sequence number from sender *j* have arrived at this member. Each member also maintains another **n**-element "Live" array **L** reflecting current group membership.

The most intuitive way to detect stability is for each member to send its sequence number array **R** to one designated member, the co-ordinator. After receiving the sequence number arrays from all the members, the co-ordinator generates a stability array **S** where S[j] is the minimum of the *j*-th element of every member's sequence number array. The co-ordinator then multicasts the stability array **S** in the group.

When the group size is large, an implosion problem will occur at the co-ordinator, which makes the naive method not scalable. Adding a multi-level hierarchy will reduce the implosion problem but introduces new problems. One such problem appears when some interior node in the hierarchy crash. Also in a large multicast group, membership change is frequent, requiring the hierarchy to be rebuilt frequently. This makes the pure hierarchy not so scalable. Therefore three design features need to be taken into consideration for GSGC to be scalable:

- the implosion problem needs to be eliminated
- traffic load generated by the protocol needs to be minimised
- the state information passed around cannot grow proportionally with the group size

The above can be achieved by using a *gossip* technique. The protocol is divided into equally timed steps. During each step, every member constructs a gossip sub-group based on their location on the network. Each local group has a stability controllers (SCs). The protocol proceeds in two phases. In the first phase, each member gossips to other members in its local group trying to obtain stability information within the local group. After the local stability arrays are constructed, the SCs start the second phase by gossiping among all the SCs. After one SC receives the stability information from SCs representing the other local groups, the global stability array is constructed and multicast to the entire group.

- **Summary**

After providing a brief overview of the design principles of HGCP and GSGC, which can be applied to network layer reliable multicast transport, we believe that HGCP's design principle is a better solution. In order to achieve stable group membership information and detect failures in a large session, GSGC is a better solution. However, this does not deal with congestion control. Whereas, HGCP deals with congestion control but if a group manager in a *cotery* is down, another child node needs to be become a group manager very quickly. The other issue to consider here is that in order to apply GSGC's design to a network protocol, the existing protocol needs to be modified, whereas, RMTP already follows a very similar pattern that is presented in HGCP which is hierarchy based. This helps us to conclude that RMTP at this stage meets most of the criteria that is required for conference control from network perspective.

5.7 Conclusion

In this chapter, reliable multicast transport protocol requirements for conference control, and floor control have been reviewed. There are protocols like RMTP/STORM, NTE and SRM which are designed for specific applications. SRM is a robust protocol which meets a lot of the requirements for conference control and can provide scalability for CCCS framework. MTP and RMTP meet certain criterias too. However, these protocols need a level of customisation or a level of adaptation to be ideal protocol for conference control. This chapter also looks at the limitation of these protocols and the limitation of floor control to achieve scalability. Therefore, if a reliable multicast has to be designed to meet the requirements of floor control and all the other facets of CCCS framework, it can be quite complicated to cater for ordering, congestion control, pre traffic flow etc. In order to keep it simple, we need a mechanism where the status of the floor holders is multicast in every few seconds to the group. If a user wish to bid for the floor, the request is multicast too. The stabilising time/converging time grows as the number of participants grow normally, so a hierarchical system will be a better solution. It is also required to provide a distributed model for retransmission and keep the status of receivers up to date.

Chapter 6

A charging model for Sessions on the Internet

This chapter describes a billing and charging technique that can be used as a part of conference control. In order to manager resources between multiple users in a multimedia conference on the Internet, a charging policy can be an effective solution. In this chapter we propose a model to show how and why that is the case.

One idea behind Common Conference Control Services is to identify missing components of different conferencing architectures and to find common grounds that can unify them. The conferencing architectures that are analysed in this thesis are all missing a charging mechanism. Also, at the same time, charging users for services provided by any video conferencing protocol can be a common policy. Therefore, in this chapter, a new charging model is proposed and analysed that can be a common charging model for any conferencing architecture. This charging technique can be integrated in any conference control protocol where users joining a conference over the Internet can be billed for their usage of the facility.

A chargeable session, which could range from high-speed web browsing to real-time conferencing on the Internet, may consist of more than one underlying chargeable service. Typically there will be two, one at the network layer and one at the session layer. Since different applications can have different demands from the network, a generic charging scheme has to separate the service provided by the network from the service provided by an application/service provider. In this chapter a pricing model is proposed which is session based and the impact of this on real-time multimedia conferencing over the Internet. In this model, we are trying to allow for the optional integration of charging at the network layer with charging at the session layer, while keeping the underlying technologies still cleanly separated. Adding such a charging mechanism as a value added service in the basic framework of CCCS shows the extensibility of the generic architecture for conference control.

In this chapter, a session based charging is proposed, *we use the term "session" to define the lifetime of activities of a single/group of users*[8]. The aim is to provide protocol independence, in the sense that, different sessions (e.g. multimedia conferencing, multiplayer games or e-commerce activities) from the application layer can be charged independently from any different basis for charging for network resources. This chapter also highlights the fact that the main problem with pricing application on the Internet is not just a simple case of analysing the most technically feasible pricing mechanism but also of making the solution acceptable to users. We take the position that session based charging is easier for end users to accept and understand and show why this is the case in this chapter.

6.1 Problem Scenario

As the popularity of the Internet grows, the number of services offered over the Internet grows with it. Users normally pay a flat fee to obtain Internet access, and are forced to get used to a service which is not guaranteed and which is prone to variable delays. However, different applications have very different service requirements. For instance, some applications like email, can tolerate significant delay without users experiencing discernible performance degradation, while other applications, such as audio and packetised video degrade perceptibly even with extremely small delays [Cocchi93]. With rapidly diverging types of tasks, the need for traffic characterization on the Internet is becoming very obvious. Cocchi et al [Cocchi93] have argued that, in order to produce performance incentives, it is necessary to support service-class sensitive pricing for any multi-class service discipline. Using this paradigm it is possible for users to prioritise their applications to conform to what they perceive to be acceptable QoS values. In this situation the user has an option to pay a higher price for higher quality.

[8] It is a natural consequence of the definition of state in CCCS

Services on the Internet have a two level matrix for charging. One is from the application perspective, and the other is from the network perspective. Research has been done to provide better than best-effort QoS in the network and to provide a corresponding charging model for the added QoS (e.g., charge for throughput, bandwidth, delay etc.). Whereas, application related pricing, i.e., charging a certain fee for an application has been left to different application/service providers. These applications can have either a fixed fee or can be usage based (i.e. charge the users if they have used the application over a certain time period). In this paper, *the term "pricing" is used to refer to the process of setting a price on a service, a product, or on content.* Whereas, "*charging" determines the process of calculating the cost of a resource by using the price for a given record, i.e. it defines a function which translates technical values into monetary units* [Stiller98].
As previously mentioned, different applications can have very different demands from the network. Therefore, in order to provide a comprehensive service for an application, a user must be able to deal with separate charges for both the network and the application QoS. For example, in a video conference, participants may just want to listen to a conference and may not require a guaranteed bandwidth. In this case, these users can be charged to join the conference (e.g. to obtain the password to join the conference) but pay nothing for reserving network resources. However, if the network resource is scarce then a price will combine both the (minimum amount of) network resource required to transmit the conference and the facility to join the conference(obtaining password – an access key). To summarise, it is best to provide a charging scheme that is not directly integrated with network QoS and specific applications.

A number of approaches have been proposed for control of usage and explicit allocation of resources among users in time of overload, both in the Internet and in other packet networks [Clark95 (a)]. RSVP [RSVP], in combination with the Integrated Service model, can be used to explicitly reserve a path or flow between end points in a network. Recent research has focused on a more generalised means of providing network QoS based on tagging packets where 'out' tagged packets receive congestion indication first and will be dropped when congestion occurs (diff-serv)[Clark95 (a)]. The goal of session based pricing is to allow an Internet Service Provider (ISP) to charge for applications that can use a variety of network reservation mechanism; such as RSVP, diff-serv, or DRP [White98].

Note that, this chapter initially emphasises why charging is a useful service for multimedia sessions such as conferencing on the Internet. It also explains the uniqueness of session based charging. The chapter then points out how a value added service like charging can be easily implemented in the CCCS framework to demonstrate its extensibility.

6.2 Review of basis for charging in the network

The main reasons for charging on the Internet are:

- to cover the cost for providing the service by service providers
- to make a profit (providers)
- to control the behaviour of users or limit the usage to benefit higher paid traffic.

Different mechanisms have different types of technical and economical advantages and disadvantages. In [Cos79], it was shown that users reduced their usage of the network when faced with usage-based charging. The complexities of understanding the criteria the users are paying for have an affect on payment as well. That is to say, if a user is presented with a complex bill that shows different criterias, and how different schemes they have subscribed to have different prices, there is a likelihood the user will prefer the flat -rate option.

The charging policy in telephone networks has existed for a long time and works very well. Telephone companies offer a menu of local calling plans, some usage-based (e.g., metered service), some capacity based (e.g. unlimited service), and some a combination of both (e.g. a certain number of free minutes per month, plus a metered rate for calls in excess of this number). It is likely that the same will happen in computer networks, with some users choosing usage based and others choosing capacity based charges, and many being somewhere in- between [Shenker96].

The two most discussed pricing schemes which can be implemented vary easily for the Internet traffic are:

- Capacity pricing
- Usage based pricing.

In **capacity based pricing**, a user would purchase a profile, called an expected capacity profile, based on the general nature of his/her usage. For example, a user exploring the web would need a very different profile from a scientist transferring a sequence of large data sets [Clark95 (b)].

Expected capacity pricing has the advantage of stable budgeting for network use. Also, expected capacity gives providers a more stable model of capacity planning. If users are permitted to install and use different profiles on demand, the provider must provision somewhat more conservatively, to deal with peaks in demand. However, the biggest drawback of this scheme is that to this point the description of bandwidth allocation has been in terms of the sender of the data, when the sender may be generating data because the receiver initiated it (e.g. in ftp case, where the server may be sending data when the user has requested the file).

In **usage based pricing**, the users pay for the volume of traffic (as well as length of time) they are interested in. The argument could be that if the resource is limited and the existing resources are used in different ways, service classes could be applied to differentiate its use appropriately. The biggest argument against this scheme is that usage based charges change the user perception and may decrease user's usage.

TCP Based pricing Edell et al [Edell95] have demonstrated a charging system based around TCP. This system charges for bandwidth but triggers who to charge per TCP connection. It does not reflect congestion costs as the pricing information is based on time of day rather than actual network loading. The authors claim it should work for UDP. UDP and TCP impose different traffic flows on the network and it is not clear how this will be reflected in the pricing structure. Oeschlin et al [Oec98] in MulTCP modified the behaviour of TCP which reflects congestion based billing which does not work easily for constant rate traffic. Also users or software developers can get round MulTCP charging by opening multiple TCP connections to achieve the same end.

Edge Pricing Shenker et al [Shenker96] have suggested a method to price the traffic where congestion costs are estimated using the expected congestion (e.g. time of day) along the expected path. Therefore, the resulting prices can be determined and charges are assessed locally at the access point (i.e. the edge of the provider's network where the user's packet enters), rather than computed in a distributed fashion along the entire path. Edge pricing has the attractive property that all pricing is done locally. Interconnection here involves the network providers purchasing services from each other in the same manner that regular users purchase service [Shenker96].

Paris Metro Pricing (PMP) Another way to deal with congestion in packet networks is provided by the PMP model [Odl97]. Odlyzko suggests that an end-user should be required to pay more to use a particular queue, although its architecture would be identical to a cheaper queue. The idea is that the queue that is more highly priced would attract less traffic and therefore suffer from less congestion than the queue with the lower price. PMP does not deal with more than one dimension of QoS. There would need to be a number of bands for each combination of bandwidth differentiation, latency differentiation and reliability differentiation. It is not true that all high bandwidth applications also need high reliability and low latency.

Smart Market proposal : One of the most ambitious pricing proposals for best effort traffic is the "smart-market" proposal for Mackie-Jason and Varian described in [Mackie95]. In this scheme, each packet carries a "bid" in the packet header; packets are given service at each router if their bids exceed some threshold, and each served packet is charged this threshold price regardless of the packet's bid. This threshold price can be thought of as the highest rejected bid; having the packet pay this price is akin to having it pay the congestion cost of denying service to a rejected packet. There are several problems with this proposal [Shenker96]. The biggest problem associated with this scheme is that submitting a losing bid will typically lead to some unknown amount of delay (since the packet will be retransmitted at a later time), so the bid must reflect how much utility loss this delay would produce rather than the valuation of service itself. The other problem is the bid is on a per-packet basis, yet many applications involve a sequence of packets. It is impossible to independently set the valuation of a single packet in a file transfer, when the true valuation is for the set of packets.

Table 8. Summary of advantages and disadvantages of some basis of charging

Name of pricing scheme	Payment for	Pros*	Cons	Technical aspect
Smart Market	Pay for speed	Provide user with the optimal price	Would be worth only when the network is congested	Difficult to implement
PMP	Different queue priority	Simple model, traffic will get through	Multiple profiles have to be defined at each differently congested bottleneck, doesnt provide different dimensions of QoS	Simple to implement if traffic is not traversing too many congested bottlenecks
Quota	Quota of usage	Easy to establish long term contract	Sender based/ need a priori knowledge of how busy the network is	Relatively easy to implement
Usage	Time of connection	Better than flat-fee, incentive not to use the network for too long	The basis of the bill can be very variant (e.g. duration, amount of resources, no. of cells, priority etc.), disincentive to use the network at all	Depending on what is being charged for implementation can be very difficult. Multicast traffic billing can be very difficult.
Session based	Session (e.g. application)	Simple and effective,easier itemised bill for users.	A lot of market research is required to set a suitable, profitable session price	Simple from session layer but network necessarily has no handle on sessions. So in the context of networking, session charging introduces huge complexity
TCP	Bandwidth	Most traffic on the Internet uses TCP, so it has a huge customer base	Cannot use for other traffic	Technically very easy to implement.

* pros – mainly economical advantages, not technical

6.3 Entities that bundle services

When a user chooses a session based charging model, they can be offered different services with different ways to pay for them. Bundling is the service aggregation between different entities. It is a business choice made by the service (provider), made for commercial reasons (e.g. profit opportunity or simply wanting to offer a more useful service). As shown in Figure 31, the transmission service and information service are bundled together (marked as host bundling)where the user pays a certain fee to the information service provider (for example, for downloading a video), who in turn pays the network provider for providing the transmission facility. The user is not aware that the information provider is paying a fee for the transmission service.

From a research perspective, there are mainly two types of users: advanced users and novice users [Bouch98]. In the former case, users tend to have theoretical knowledge of networking environments, and are familiar with syntactic aspects applicable to real-time and data-driven tasks. In the latter case, novice users are mainly the type who have little or no theoretical knowledge of networking environments, and are unable to directly map technical syntax onto underlying conceptual consideration.

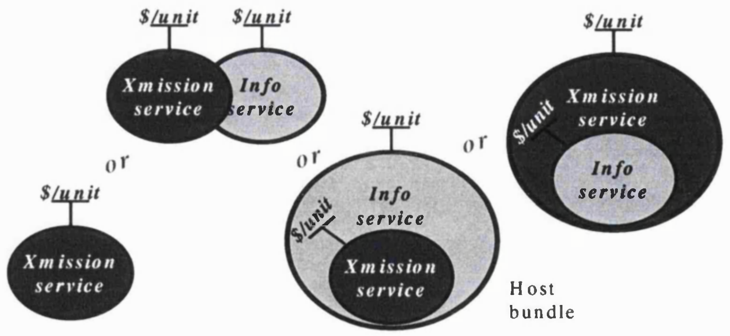


Figure 31: Bundling services

These types of users in the target market make a difference as to how services are bundled.

The framework for charging for usage of different services with different quality on the Internet is quite complex. The parameters charged for can be very dynamic and variable. For example, in the telephone system, all calls require the same network capacity and the same quality of service, whereas flows in the Internet can differ widely in their need for capacity, latency control, or other features. Especially in the context of associating value with enhanced services, it must be possible for the users to describe the service they require. The features that can be charged for are: throughput, speed, accuracy(assertions connecting QoS to the ability of the search engine for example to deliver the requested information), accessibility and reliability. Therefore, for the novice type of users, there may be a requirement to set the session price for them, otherwise they have to be educated through the process of quality of service aspects of the Internet.

In this section we look at different service aggregation methods, i.e. various ways a user can be billed for a particular service he/she used over the Internet. These mechanisms have to be easily understood by the people so that they will be interested in using them. We would like to propose that there will be mainly two ways to bundle.

There are:

1. ISP bundling
2. Session owner bundling

And there is always another approach which is based on

3. User's choice

1. ISP bundling – In this scenario, the ISPs will set a price for a given session and the hosts and the participants will directly pay their ISP for that session. This is probably not a very attractive option for the ISPs because they have to work out separate prices for interconnecting with different providers for each session, how many people possibly want that service and work out a set price for every session they are providing. ISPs providing Internet telephony services should pay access charges to the local telephone companies as do other long-distance service providers.

However, the ISPs can actually make a sufficient amount of profit by providing a price on a session basis, because a lot of users/hosts do not actually want to go through the trouble of working out a price for a session. In order to bill the users for that session, the ISP has to take into consideration that users may pay into their account (which may/may not exist , so for every session they have to create a separate billing account) or by credit card or by e-cash. As mentioned that it may not be the most attractive option for ISPs.

ISPs will have policies which can be exchanged among policy-enabled entities. DIAMETER [Rubens98] is currently a proposal which for example, can be used for ISP bundling. It is designed as a common platform for several Internet services, such as AAA (Authentication, Authorization and Accounting), network-edge resource management and VPN (virtual private network). So for example, when a caller (e.g. a SIP proxy server) is being notified to set up a call for a user, it first initiates a DIAMETER request command to its policy server with all the information about the user. The server, in turn, checks the request against an admission control policy database, and returns the findings in a DIAMETER response message [Pan98]. [Pan98] attempts to cater only for ISP bundling, but it is unlikely DIAMETER would be the preferred solution for non-ISP bundling. Therefore a more general solution would be beneficial.

2. Session owner bundling – In this scenario, the master of ceremonies (e.g. an organiser of a conference) sets the price for individual user or mainly an organization, where the novice users do not have to know the implications. For example, user A decides to host a conference in UCL, 1999 for 2 days. This conference requires access to the Mbone in order to multicast the session. So the host has to work out what is the minimum bandwidth required to transmit video and the minimum bandwidth required to transmit audio are. After that the session owner sets a common price that absorbs and hides peaks and throughs in costs for each participant. A slight premium allowance above the expected average cost involved underwrites the host's risk. This might either turn a profit for the host or be returned to all participants in equal shares (co-op dividend). Each participant's cost to the host will depend on their ISP's price,

but the host is wholesaling (hiding) this to participants. This may be a lot of work for the host to work out a suitable price. This scheme will be attractive for a type of host who holds a lot of sessions like that a year and the host is likely to be a big organization. As for the user is concerned, they do not have to worry about the technical aspects of the conference and it makes it definitely simple for them to just pay the host and participate in the conference. The question remains, will a user be interested in paying a fixed amount for which they are confined to the policy the host/session owner has set?

3. User - In this scenario, the user has the choice to go either with the "best-effort service" for a session or can pay their ISP directly for guaranteed service. Normally for all of the above option as well, the frame rate for video and for audio the required bandwidth will be advertised on SDR(see section 6.4 for further discussion). Therefore, it is up to the user to pay a certain fee for a certain amount of guaranteed service. For novice users, they do not necessarily need to know the technical details. There will be the option in the form of a sliding bar marked with values (either monetary values or other forms of prices), and increasing the value of the sliding bar will increase the quality.

   With this option, the host or the ISP do not have to set certain prices for everyone for different sessions. Also, it gives the user the flexibility to go with their own policy, i.e. they are not confined to ISP's or the host's policies.

For all of the models of payments above strong security is necessary both between routers and policy servers and between policy servers and the billing system that connects policies to economics because their interaction implies financial transactions. Whatever the bundling scenario is and whether an ISP or a user is setting the price, they can use a session based pricing interface (as discussed in the section below) to serve their purpose.

6.3.1 Matrix for charging sessions on the Internet

Here are some matrices for charging services. Let us assume **S1** and **S2** are two arbitrary (chargeable) services. In this case one will be for the network and the other one for the session hosting. **U** is the user and **A** is an agency. The matrix to define and combine the two services has some issues like:

a) What is being paid for
b) By what you are paying for (the basis)
c) By whom each service is paid

Let us imagine a scenario where user **U** has to pay agency **A** for a distance learning session. **A** has bundled **S1** and **S2** together. User **U** pays **A** for both joining the conference (i.e. **S2**, the hosted service) and the usage of the network (**S1**). Although **U** may not be aware of these details. **S1** can be a rate fee for the usage of the time or volume of traffic generated on the network; although if the session takes place at a busy period the rate can be different from the non-busy period. The fee charged for **S2** can either go to the organisers of the session, the content provider or to a different agency, who pays his providers. As it can be seen in the Figure below, the participants pay the agency who pays the providers. The providers have to meet their service level agreement or otherwise they may be penalised.

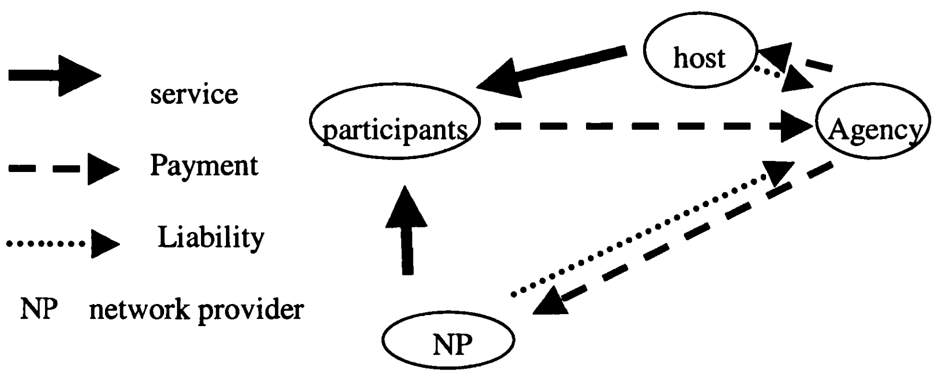


Figure 32: Model of interaction between participants and agency

The billing and charging mechanism can be very complex depending on policies and regulations. Charging for multimedia sessions such as conferencing and the complications associated with it has not been researched on extensively at the time of writing. Session based charging model can make this process a lot simpler. In the following section an attempt will be made to provide an overview of this model and its uniqueness.

6.4 Model for application driven session pricing

This section proposes a possible example of a user interface and a model for session based charging where RSVP is used to reserve the underlying resources (in section 6.4.2) that could be used for any of the bundling scenarios discussed in section 6.3. An important aspect of the problem of designing a model to charge for real-time applications on the Internet is that the Internet architecture is based on the network layer not knowing the properties of the applications implemented above it. Therefore, in this model knowledge of the underlying resource management and of the network implications of providing a guaranteed service is not necessary and has been separated from the applications. ISPs or the bandwidth broker sets a certain price for each session that can be accessed from the session layer. While in this chapter and in this model we have focused on monetary values to participate in a session, the underlying accounting structure and pricing architecture should allow the use of other incentive forms if they are locally applicable.

The design philosophy of this model is quite simple. Let us take a multimedia conference for example, there will be few participants among which some are just listeners. This session is advertised by some arbitrary means (e.g. SDR [Kirst97] or a web page), with the session's price being fixed a priori. As discussed in section 6.3, this follows a user driven system where the user has the choice of either paying a certain fee for guaranteed QoS or not paying. So there will be an "agent" who will be responsible for collecting the payment. The session based pricing comprises a "back-end", whose job is to inform the service provider or the initiator (depending on who is charging and what the policy is) that the specific session is being paid for and a guarantee for that service for that price is required. Each router, on receiving a packet, must be able to determine whether the router is within the paid region. There are only two ways that a router can have access to information about a flow. Either it is stored in the router (this is not the preferred option), or in the packets of the flow.

The second part is a "front-end" which allows a client to provide inputs in the selection process. In this scenario we have used multimedia conferencing as an example where there are different classes of participants. So the participants who are just listeners can choose to pay a flat fee whereas a speaker will pay an additional amount for that session. However, for example, if the speaker is an invited speaker then he/she may pay nothing. Floor control [Kausar (c)] for the session plays a very useful part for this pricing scheme.

If a user initially chose the option not to speak then the floor control option is not enabled. However, we realized that a listener may have a question at the end of a session, but that the amount of traffic that will be generated by this question may have an impact on the network if the resource available is scarce. For most of the existing conference tools there is a facility to use a text based "chat" option where the users can enter their questions in text format rather than using audio which generates more traffic. A possible example of front end could look as shown in Figure 33:

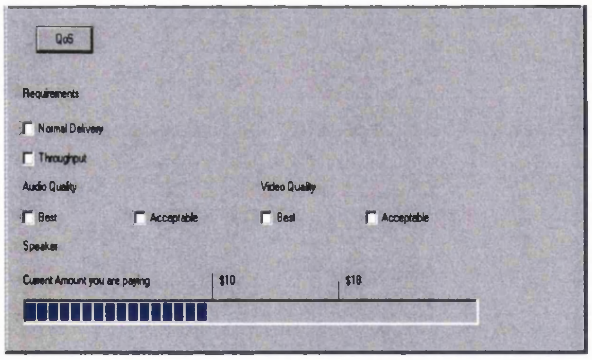
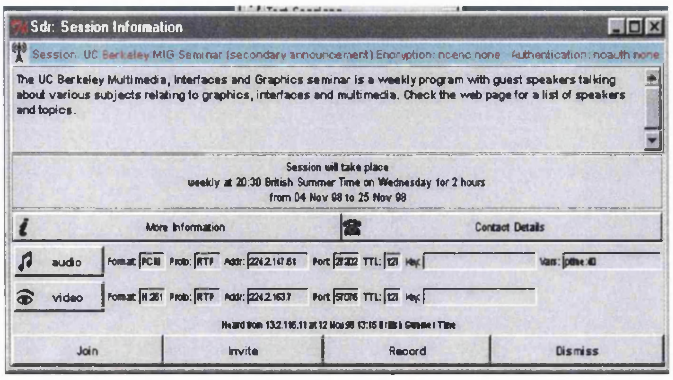


Figure 33: An example of front end of payment service

An Mbone session directory SDR [Kirst97] is used to advertise multimedia conferences, and to communicate the session addresses (whether multicast or unicast)and conference-tool- specific information necessary for participation. This would be an ideal tool to advertise the prices associated with the sessions. Currently the user interface appears as shown in Figure 34. An extra option with QoS details which include a sliding bar for payment can be added to enhance the features of SDR.


Sdr: Session Information

Session: UC Berkeley MIG Seminar (secondary announcement) Encryption: noenc none Authentication: noauth none

The UC Berkeley Multimedia, Interfaces and Graphics seminar is a weekly program with guest speakers talking about various subjects relating to graphics, interfaces and multimedia. Check the web page for a list of speakers and topics.

Session will take place
weekly at 20:30 British Summer Time on Wednesday for 2 hours
from 04 Nov 98 to 25 Nov 98

audio Format: PCM Proto: RTP Addr: 224.2.147.61 Port: 28202 TTL: 127 Key: Vars: ptime:40

video Format: H.261 Proto: RTP Addr: 224.2.163.7 Port: 57076 TTL: 127 Key:

Heard from 132.116.11 at 12 Nov 98 13:15 British Summer Time

Figure 34: Session directory would need an option for payment

6.4.1 Support from Network layer

The Internet today offers a single class of service, where all packets are serviced on a best effort, First-in–First-Out (FIFO) basis. Disrupted audio and video due to packet losses make multimedia conferencing less efficient and less effective. The applications that generate traffic can be located on a continuum (see Figure 35) which also represents the delay tolerance. As the amount of real-time traffic increases there may be a corresponding need to define a richer set of QoS parameters for these traffic types [Bouch98].

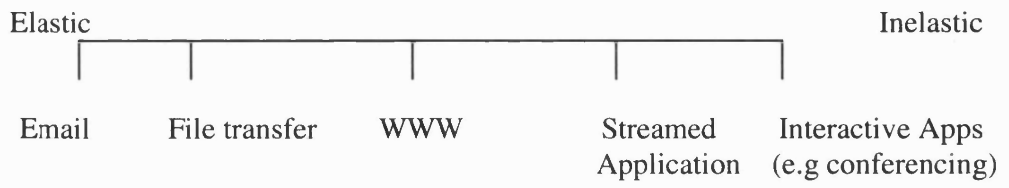


Figure 35: Relative traffic elasticity

Although users are normally prepared to put up with delay with email because it is expected to be delivered later in the day and picked up some other time, one may send an urgent email which can be treated as a real-time or inelastic application (for example, an email informing someone to join a conference immediately).

In order to guarantee the service that is chosen from the session based application pricing interface, the network has to provide enough resources. Since an RSVP API [Stevens (b)] is currently available, we suggest integrating RSVP with the different models of session based pricing. However, as discussed in section 6.2 there are other ways to reserve the resources or characterise the packets that are being paid for in network layer. In this thesis, we are not focussing on any particular charging scheme for network or service, any number of combinations can be used. We are assuming the commercial application will be paid for and the underlying resources will be reserved or characterized in a way that will support the application.

To ensure voice and data are being delivered properly, users can make the use of end-to-end resource reservation protocols to set up reserved "flows". Another alternative is to mark the packet header as "premium service" so that they can be delivered with low delay and rate guarantees inside the network. Both approaches imply that the network-edge routers may need to interface with policy servers to manage link resources. Although as shown in Table 8, session based charging proposed here, has the advantage of hiding all the underlying details from the user and has a better chance of being accepted by them; it is necessary to look at a model which would reserve resources at the lower layer to filter up to the session layer pricing. In next section RSVP in conjunction with session based charging model is discussed.

6.4.2 Applying session based charging model

The session based charging model is conceptually compatible with the layered network model. The separation between sessions and network layer pricing is a new architectural consideration. This section briefly describes how conference control and reservation of network resources will provide an overall solution for session based charging model. In here, Fankhauser's [Fankhauser98] proposal for (network) resource reservation based charging has been used as an example of a useful tool that will help to give an overview of applying session based charging model in a "real" network. Frankhauser's implementation can be used in conjunction with session based charging which shows how the consumption of resources at the lower layer would filter up to the session layer pricing.

It is assumed here that, ISPs or the bandwidth broker sets a certain price for users for each session which will provide a type of service class (it could range from minimum delay to highest throughput) for that price. In order to guarantee the underlying resources that could provide that sort of service, it is necessary to reserve those resources. In Fankhauser's implementation a simplified version of RSVP has been used for resource reservation and it uses flows as basic units of charging and accounting. This modified RSVP in conjunction with session based charging mechanism proposed here in a conference control model provide an example of a total solution for commercial model of conferencing.

In this model, the price is calculated dynamically according to the load on the network, although the base prices are set by the operator at each node. *The basic unit sold via a session based charging interface to the user is a bandwidth reservation over a fixed period of time.* Besides the base-pricing, which has to be determined on grounds of business and strategic market decisions, the model provides options which control the price when congestion is about to occur. The options set parameters which are expressed as link-load factor and can be used adaptively to maximise utilization of the resource and to minimise the number of rejected reservations without charging the base price.

In Fankhauser's implementation RSVP messages used for reservation setup, such as PATH and RESV messages, are enhanced by adding payment information and price queries. Each network node along a transmission path features the basic functionality of an Integrated Service Router (ISR) which is also the case for sending and receiving hosts with applications that use the extended RSVP API (application programming Interface) to communicate pricing information. RSVP messages are forwarded through a socket connection to the RSVP daemon running in user space. The enhanced admission control checks pricing information when reservations are made. The integration of reservation and charging protocols fit very well and have several interesting and attractive properties:

- Periods of reserved bandwidth can be accounted and charged at the current market price at each router by using the flow specification (flowspec). Once a flow has been accepted and the resource allocated, traffic control mechanisms (classification and scheduling) ensure that the requested bandwidth is allocated to the flow.
- RSVP messages are processed and forwarded hop-by-hop. This method enables every provider to collect money for its own resources. No inter-provider agreements are needed.

Charging messages: for Frankhauser's simplified version of RSVP the following messages are used and modified with payments and other charging relevant information:

- **PATH** messages are used to pin a path and setup a state to make sure that RESV messages follow the same route back to the sender. PATH messages may contain a request quote (QRQ) market prices or a sender provided payment (S_PAY).
- **RESV** messages are used to request a reservation (receiver initiated). They may carry quote (QTE) messages or receiver payment (R_PAY).

In Figure 36, the modified packet format for RSVP is shown. The "msg type" field could be any of PATH, RESV, RESVCONF, PATHTEAR and RESVTEAR. C&A flag stands for charging and accounting flags which could be QRQ/QTE, S_PAY and R_PAY.

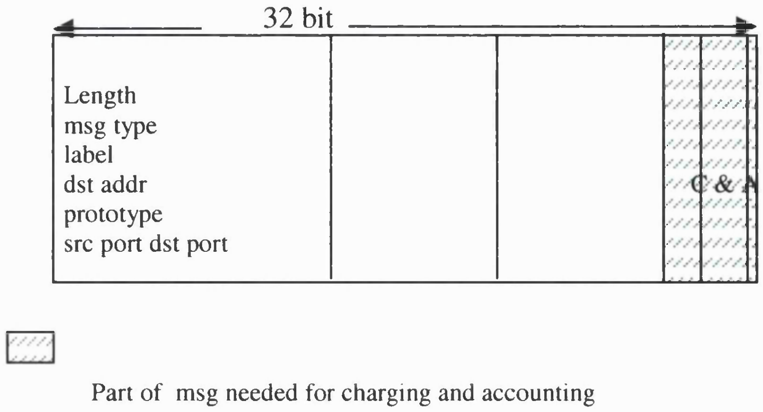
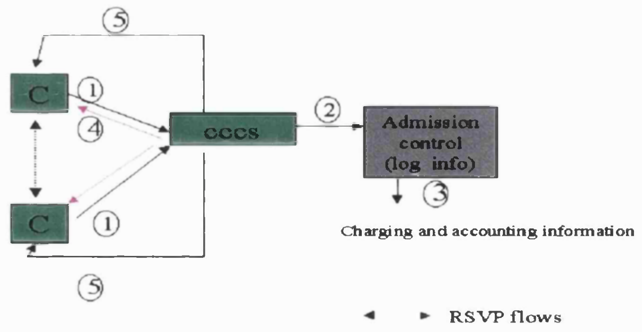


Figure 36: PDU for resource reservations enhanced by charging and accounting data

Figure 37 shows a conference control protocol that can be associated with charging. The nodes marked as **C** are clients. The sequence of actions will be as following:

1. Clients register themselves with the CCCS (or any other conference control entity).
2. Their ID (for example the port number, IP address) is logged. The logged information is passed to the admission control policy.
3. The admission control policy can be associated with Fankhauser's implementation as discussed above. This admission control policy confirms whether the client logged is an authorised user to be granted guaranteed quality of service from the network for the duration of the conference. If the client is authorised the resources required for this session are reserved.
4. The CCCS notifies the client whether the client will be billed for quality assured delivery of conference.
5. Once the session is finished the charging Daemon produces a bill which could be distributed to the clients from the conference control server.



5
C
1
4
CCCS
2
Admission control (log info)
3
C
1
Charging and accounting information
5
RSVP flows

Figure 37: Conference control and session based charging with RSVP

6.5 ISP's perspective

One of the attractive schemes which perhaps allows a great encapsulation and therefore compact characterization of application specific QoS parameters is known as 'User Share Differentiation' (USD) [Wang97]. USD involves, not the separate reservation of bandwidth for each flow per session but the sharing of a pool of bandwidth among multiple users. The user is given a minimum amount of individual bandwidth, according to the user-ISP contract, and a minimum share of bandwidth over this bandwidth. It is argued that this scheme strikes the correct balance between aggregation and isolation of sources. Its additional benefits may be:

- The definition of 'user' is flexible: this implies that the level of aggregation of traffic is flexible. For example, the ISP is free to implement multiple classes of traffic to reflect the needs of different users; some users may only require a best-effort service.
- A hierarchical management structure is provided: The ISP allocates bandwidth to the user, the user then allocates among its applications. The user can choose to mark its applications to reflect loss or delay priorities. This has important implications for traffic classifications at different levels of the market structure [Bouch98].
- Incentives are provided for users to control their traffic sending rates. This fits perfectly well with "user bundling" system described in section 6.3.

6.6 Multicast Model

It is held that multicast offers significant advantages to the Internet community. Multimedia real-time applications which are being multicast pose more of a challenge to be priced and different access rates need to be considered carefully when pricing the senders/receivers. A multicast address is merely a logical name, and by itself conveys no geographic or provider information. Multicast routing identifies the next hop along the path for packets arriving at an interface, multicast routing does not identify the rest of the tree. Thus, estimating costs in the multicast case requires an additional piece of accounting infrastructure. One approach for charging the receivers is to introduce a new form of control message – an accounting message – that would be initiated when the receiver sends its multicast join message [Shenker96]. These accounting messages would be forwarded along the reverse trees towards each source, recording the "cost" of each link it traversed and summing costs when branches merged.

With the User bundling scenario, the session based pricing solves the problem of charging receiver/sender in a multicast session. As discussed previously, the user can pay a set price regardless their position in a multicast tree. If the receiver wishes to receive a session with a certain guarantee, they just have to pay. In the user bundling system, the price to be paid for a session's quality is up to the user, so whether the user is a multicast receiver or not, does not really affect the charging scheme. If the multicast tree is organised in a hierarchical structure, then the host or the ISP (if it is a host or ISP bundling system being used to pay for services) can negotiate or set a price for a particular branch of the tree. Then, if one of the child nodes joins a session which needs to be paid for, the node can obtain the price from the nearest parent node.

Another issue to be addressed is: to which party (content provider, ISP or receiver) does multicast transport offer the most intrinsic value compared with unicast transport? In overall, one can say that multicast access and peering agreements are likely to be placed on a very different financial basis from the existing unicast agreements. The figures below compare multicast and unicast data delivery, for a simple case in which both the content provider and subscribers buy access from the same ISP.

As seen in Figure 38, the multicast sender (e.g. content provider) benefits greatly from multicast, since access costs are drastically reduced. There is little multicast benefit to the receiver. To the receiver it makes little difference whether multicast or unicast is used (assuming, that received bandwidth is charged at the same rate whether unicast or multicast). By default, the ISP should charge multicast senders (e.g. content providers) more for multicast access bandwidth (sent into the network) than for unicast access bandwidth.

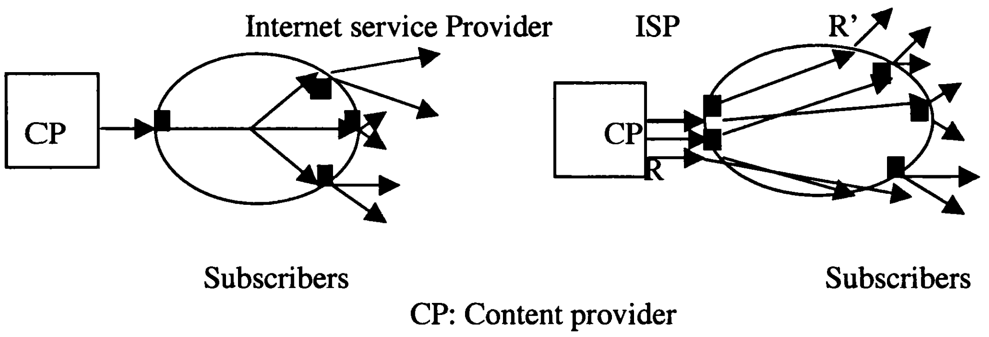


Figure 38: Comparison of (a) multicast and (b) unicast delivery

If the multicast sender is charged more, the increase in access bandwidth tariff should be in some way be related to the degree of replication (actual, average etc.) performed by the network, but should be less than would have been charged for unicast access to N clients. One of the main difficulties with charging multicast senders according to the degree of replication is that it is likely to be a considerable overhead for the ISP to measure the actual degree of replication on a per-session basis. If the multicast access tariff for senders is based on an average degree of replication (averaged across sessions), then this will not cater for different ranges (tens to thousands of participants).

6.8 Conclusion

Different types of traffic sent into the network may have different QoS requirement associated with them. The satisfaction a network user derives from their network access depends on the nature of the application being used and the quality of service received from the network. Since the nature of the Internet architecture is based on the network layer not knowing the properties of the applications implemented above it, we have proposed a session based charging model that operates over existing network reservation/pricing schemas and augments them to take into consideration additional needs of applications.

Thus, we view existing network reservation/charging schemas as providing a baseline set of services to a user. Subsequent or value-added services and refinements of the network - services are accomplished with a session-based pricing schema. One example of its realisation could be in SDR, which is aimed at users and thus can provide a simple and straightforward way of conveying price-to-function relationship.

Having explained the benefits of the session based charging model, it was demonstrated how this value added service could be added to a conference control framework. A generic model for conference control such as the CCCS can easily accommodate this facility. An overview of how the consumption of resources at the lower layer would filter up to the session layer pricing has been given by combining session based charging with network resource reservation.

Setting a session price that will profit the ISP or the content provider, and yet still be price-competitive with their competition, can be difficult to predict. The complexity of predicting and implementing a profitable price for session based pricing is still an open issue and a subject for further research. However, session based pricing has the attractive features of providing a more direct way of communicating costs to the user and of having the flexibility to implement it with any different basis for charging network resources.

Chapter 7

Conclusion

This research is about how communication using different conference control mechanisms can be seamlessly integrated into a single mechanism. We provided a complete framework for conference control and implemented some features over the Internet to prove the concept. We analysed the requirements of this architecture from the network perspective. In the process we also highlighted the interesting research problem of charging few services that are offered by different facets of conference control and conferencing in general.

The Common Conference Control Services (CCCS) has been proposed which showed that it is possible to provide a practical system that can exist between a full co-operative and full autonomy extremes for computer based multimedia conferencing. The CCCS framework allows more flexibility than the standard bodies to date have allowed. As mentioned from the beginning, CCCS is not a specific groupware: it is a communication framework for conference control services which deals with user visible functions and internal management functions of a conference. By using state transition diagrams and formal description language it has been shown that the Main Control Services part of the CCCS provides all the mandatory functions required to participate in a conference. If a user of one protocol wants to join a session from another, the translations and mappings of various messages done by the Gateway Services of CCCS and the location of the GS servers are not visible to the user. Therefore, it can be concluded that, the main contribution from this research has been to derive a set of common conference control functions that are independent of specific application or specific network architecture.

In Chapter 4 and 5, conferencing and some conference control features have been analysed from the network-layer perspective and reliable IP multicast has been chosen to provide the facilities. It has been shown that a hierarchical protocol like RMTP or HGCP with just a few adaptations can meet most requirements for conference control. Most other IP multicast protocols deal with reliable delivery of data and provide some other essential characteristics like congestion control or ordering but not all of them are present in one protocol. It has been shown that a network layer protocol needs to provide congestion control, ordering, reliability, scalability and yet simple to implement to support conference control features effectively.

In Chapter 6, a session based charging mechanism has been proposed that can be a value added feature for conference control which can be embedded in a conference's policy. Session based charging can be integrated in the CCCS framework to show that the framework for conferencing proposed here is easily extensible. The charging scheme is positioned "on top of" other charging schemes at the network layer. After giving an overview of past work in network pricing, it is argued that charging per session can be more appealing to end users than charging per consumption of network resources. The other advantage is that the separation between session and lower layer pricing provides a useful decoupling which could make pricing more flexible. It is also highlighted that session based charging model can relate to different service bundling models and multicast.

- **Limitations**

The main limitation of this framework is, it is based on a Broker pattern where the "Broker" or the "gateway" has to be stateful to provide reliability and billing information to its clients. This model tends to accommodate only a limited number of participants. For a truly Internet based conferencing the conference control has to be based around a stateless architecture which is currently provided in the IETF model. However, in a distributed environment several gateways or the brokers will be provided, which together can accommodate a large number of participants. However, in this thesis a simulation of such a system has not been demonstrated to verify the claim.
The implementation of the CCCS has taken only one step in the direction of providing a generic distributed architecture for conference control. It has only demonstrated very few common functions like join, leave, invite and floor control when different conferencing architectures correlate. It has not shown negotiation of capabilities, security features and

interaction of different media tools that are provided in different architectures. In addition, session based charging has not been implemented as a part of the CCCS's policy.

- **Future Work**

There are several ways that future work could lead to:

1) Security: Need to provide different levels of security in conference control. In this thesis it has been assumed that all different architectures like Mbone based conferencing and H.323 family of conferencing have some minimum security features built into it. For example, a security feature will be to provide users with a password and then allow them to join a conference if they have provided the password. However, the CCCS itself does not have a security feature. So if a user from IP network is interoperating with another user on the Telephone network using the CCCS and this service is to be billed for, then the users need to provide some form of authentication to CCCS. Currently there is no mechanism to cater for that.

2) Distributed control: In Chapter 3, in order to prove the concept of CCCS as a conference controller that provides interoperability, one Gateway Services (GS) server has been used to interconnect several different types of client. The measurements taken for speed and the number of messages were also based around that model. In future, it is required to distribute the GS servers across a wider area network. A particular server in one particular area needs to be in charge of only a limited number of users. These servers can be based around the geographic locations of the clients. So for example, if five H.323 client in New York need to co-ordinate and participate in a conference with seven other Mbone based clients in UK, then it is feasible to locate one GS server in New York and one in the UK.

   These two servers need to keep local registry and co-ordinate with each other. If one of the servers crash, the other server should be able to detect that and take appropriate actions. The performance measurements taken from this type of scenario will present a more practical session where a conferencing architecture like the CCCS can be better justified.

3) Charging: The separation between sessions and network layer pricing presented in Chapter 6 is a useful new architectural consideration. Although it has been suggested that any method of reserving network resources such as RSVP can be used to assure smooth operation of applications, a practical experiment will be of significant value to research community. The consumption of resources at the lower layer filtered up to the session layer charging needs to be observed and carefully analysed.

4) IN services: At the end of Chapter 3 and in Appendix 3, a list of IN service features has been pointed out. In future, the CCCS should be able to cope with some of these IN features like call queuing, Originating Call screening (OCS) and Call Forwarding (CF). Asynchronous events like email or voicemail etc. should be able to be forwarded when different conferencing architectures correlate. So if a participant wants to invite another, and there is no response, the CCCS currently only informs the caller that the callee is not present. It has been advised that if the caller leaves a text based message for example, the CCCS should forward that message to the callee.

5) GS Location service: Currently the IPTEL working group in the IETF is looking into gateway location service which investigates a protocol for maintaining gateways and distributed call routing databases across multiple administrative domains for voice over IP services. A similar type of protocol or application needs to be designed which will find the nearest CCCS entities that will carry out both aspects of conference control functions between two or more clients. As discussed in 2), in future, it is required to distribute the CCCS server across a wider area network. A particular server in one particular area needs to be in charge of only a limited number of users. Therefore the users/applications need a "location mechanism" to find the nearest CCCS server. The selection process which may reside on distributed clients or databases will choose the CCCS entity nearest to the maximum number of clients and

the databases need to updated frequently to keep a register of all possible servers around.

- **Contributions**

In this thesis, a generic conference control framework has been presented. It is shown that a single mechanism can support interoperability between two or more innately different architectures dominating the industry and the research arena. A set of services and activities that are present in a canonical model of conferencing have been derived and using these generic set of features a gateway has been built between a tightly coupled conferencing to loosely coupled conferencing. The framework is analysed from the networks perspective and the requirements from the network are identified. The network has to support the vital services offered by the conference control framework such as reliability, congestion control, and scalability. This work also proposed what future conferencing architecture over the Internet will be like and how a generic architecture should be flexible to add these facilities. In order to manage limited resources available in the Internet, a charging scheme needs to be in place. This thesis proposes a new network layer independent charging scheme, which can be easily integrated in a generic model of conference control, proposed here to show its expandability. The objectives outlined at the set out of this thesis have been met.

Reference:

[Ahuja] S., Ensor R., "Coordination and Control of Multimedia Conferencing" IEEE communications magazine, May 1992, pp 38-43

[Armstrong] F., RFC1301, "Multicast Transport Protocol" February 1992 http://www.ietf.org/rfc/rfc1301.txt

[Bagnall] P., Poppit A. "Taxonomy of Communication Requirements for Large-scale Multicast Applications", Internet Draft , draft-ietf-lsma-requirements-01.txt, May 1998

[Boyd] J., "Floor control Policies in Multiuser Application", INTERCHI '93 adjunct proceedings, 1993, pp. 107-108

[Bormann] C., Ott J., "Simple Conference Control Protocol", Internet draft draft-ietf-mmusic-sccp-00.txt", Universitat Breman, Germany, Dec 1996

[Bouch98] Bouch A., " A User Centered Approach to the Design and Implementation of Quality of Service and Charging Mechanisms in Wide-area Networks" – 1st year report, http://www.cs.ucl.ac.uk/staff/A.Bouch

[Buschman] F., Rohnert H., Sommerland P., "Pattern Oriented Software Architecture: A system of patterns" John Wiley & sons

[Clark D] "The Design philosophy of the DARPA Internet protocols", SIGCOMM 1988 (Stanford, CA, Aug 1988), ACM

[Clark95 (a)] Clark D. "Adding Service Discrimination to the Internet " September 1995, presented at MIT workshop on Internet Economics

[Clark95 (b)] Clark D.(MIT), "A Model for Cost Allocation and Pricing in the Internet", presented at MIT workshop on Internet Economics, Mar 1995 http://www.press.umich.edu/jep/works/ClarkModel.html"

[Cocchi93] Cocchi R., Shenker S., Estrin D., Zhang L. "Pricing in Computer Networks" – Motivation, formulation and Example , IEEE/ACM Transactions on Networking, vol. 1, Dec. 1993.

[Cos79] Cosgove J., Linhart P. "Customer Choices under local Measured Telephone Service" Public utilities fortnightly, 30, pp 27-31, 1979

[Crowcroft] J., Handley M., Wakeman I., "Internetworking Multimedia", UCL press, http:// www.cs.ucl.ac.uk/staff/jon

[Deering] S., "Host Extension for IP multicasting", Stanford University, RFC 1112, IETF, August 1989

[Dommel] P., Aceves JJ (1995) "Floor Control for Activity coordination in Networked Multimedia Application" - Proc. 2nd Asian-Pacific Conference on Communications (APCC)'95, Osaka, Japan, June 12-16, 1995.

[Edell95] Edell R J., McKeowen N and Varaiya PP., "Billing users and pricing for TCP" IEEE Journal on selected areas of Communication 1995 pp 105-115

[Fankhauser98] Fankhauset B., Stiller G., Plattner B., N. Weiler, "Charging and Accounting for Integrated Internet Services - State of the Art, Problems, and Trends", In proceedings of INET '98, Geneva, Switzerland, July 1998

[Floyd S] V. Jacobson, S. McCanne, C. G. Liu, and L. Zhang (1995), "A Reliable Framework for Light-Weight Sessions and Application Level Framing" - ACM SIGCOMM '95. Boston. August 30-September 1, 1995.

[Floyd] S., Varadhan K., Estrin D (998) "Impact of Network dynamics on End-to-End protocols: Case studies in TCP and Reliable Multicast "- research draft "http://www.isi.edu/~kawnan/VINT/ic98.ps"

[Fromme] M., Pralle H. "An Address Resolution and Key Exchange Protocol for Conferencing Applications on the Internet, Confman 2.0", Interactive Distributed Multimedia Systems and Telecommunication Services, 5th International Workshop, IDMS'98, Oslo, Norway, September 1998. Proceedings

[Gary] A. Thom, "H.323: The Multimedia Communication Standard for Local Area Networks for charging", work in progress, Delta Information systems, Inc., Nashville 1998

[Guo] K., Hayden M, Robbert van Renesse, Werner Vogels and Kenneth P. Birman , Cornell University, "An Efficient Gossip-Style Garbage Collection Scheme for Scalable Reliable Multicast", work in progress December 3, 1997

[Greenburg91] S., Chang E. "Computer support for Real Time Collaborative Work", Proceedings on Numerical Mathematics and Computing, September 1992

[H.225] ITU Recommendation "Call signalling Protocols and Media Stream Packetisation for Packet Based Multimedia Communication Systems", 1998

[H.245] ITU Recommendation, "Control Protocol for Multimedia Communication"

[H.323] ITU Recommendation, "Packet Based Multimedia Communication Systems", 1998

[H.450] ITU Recommendation "Intelligent Networks: Supplementary services"

[Handley 97] "Network Text Editor (NTE): A scalable shared text editor for the Mbone", Proceedings of ACM Sigcomm 97, Canne, France, 1997

[Handley M], Thesis for PhD, "On Scaleable Internet Multimedia Conferencing Systems", University College London, UK, November 1997

[Handley] M, Wakeman I., and J. Crowcroft, "CCCP: Conference Control Channel Protocol-a scalable base for building conference control applications," *ACM Computer Communication Review*, Vol. 25, pp. 275-287, Oct. 1995.

[Handley99] M., Whelan M., Perkins C. "Session Announcement Protocol", Internet draft http://search.ietf.org/internet-drafts/draft-ietf-mmusic-sap-v2-01.txt, October 1999

[IBM94] Aldred B., Bonsall G., Mitchell H., Lambert H., "An Architecture for Multimedia Communication and Real-time Collaboration", IBM Systems Journal – Network Technologies and Systems, 1995, Vol 34, No. 3, 1995, pp-519-529

[Kausar (a)] N., Crowcroft J. "An Architecture for Conference Control functions", Data communication and voice networks Conference, SPIE Symposium ,19-22nd September 1999, Boston MA

[Kausar (b)] N., Crowcroft J. "General Conference Control Protocol", 6th IEE conference on telecommunication, 29th march – 1st april, Edinburgh, UK, 1998, pp 143-152

[Kausar (c)] N., Crowcroft J. Briscoe B., " A Charging Model for Sessions on the Internet", 4th IEEE symposium on ISCC , July 6-8th, Read Sea, Egypt, pp 32 -38

[Kausar98] Kausar N., Crowcroft J. – " Floor Control Requirements from Reliable IP Multicast" 8th IFIP Conference on High Performance Networking (HPN'98) The Millennium Push of Internet, Vienna September 21-25, 1998

[Kirst97] Kirstein P., Whelan E. "SAP - Security using Public Key Algorithms" Internet draft http://www.ietf.org/internet-drafts/draft-ietf-mmusic-sap-sec-04.txt, October 1997

[Kouvelas] I, O. Hodson, V. Hardman "Redundancy Control in Real-Time Internet Audio Conferencing", Proceedings of the 1997 International Workshop on Audio-Visual Services Over Packet Networks 15-16 September 1997

[LDAP] RFC 1777, Yeong W., Howes T., Kille S., "Lightweight Directory Access Protocol" www.umich.edu/~diysucs/ldap/doc/rfc/rfc1777.txt

[Levine] B., Aceves-JJ (1998) "A comparison of Reliable Multicast Protocols", "http://www.ucsc.edu/b.levine", Multimedia Systems (ACM/Springer), Vol. 6, No.5, August 1998.

[Lin], John C. and Paul, Sanjoy (1996), "RMTP: A Reliable Multicast Transport Protocol", IEEE INFOCOM '96, March 1996, pp. 1414-1424.

[Ma] G., "H.323 signaling and SS7 ISUP gatewaying procedure interworking", Work in Progress, expired internet draft , October 1998, draft-ma-h323-isup-gateway-00.txt

[Mackie95] Mackie-Mason J., Varian H. "Pricing the Internet" – In brian kahin and James Keller, editors, Public access to the Internet. Prentic –Hall, New Jersey 1995, URL: ftp://gopher.econ.lsa.umich.edu/pub/papers/Pricing_the_Internet.ps.Z

[Mbone] Mbone information site, http://www.mbone.com/techinfo/

[McCanne] S., and Jacobson, V., "Vic :A Flexible Framework for Packet Video", ACM Multimedia, November 1995, San Francisco, CA, pp. 511-522.

[Odl97] Odlyzko A. "A Modest Proposal for Preventing Internet Congestion" 1997, http://www.research.att.com/~amo/doc/recent.html

[Oec98] Oechslin P., Crowcroft J. "Weighted Proportionally Fair Differentaited Service TCP", accepted for ACM CCR, 1998

[Ott J] "A Multipoint Data Communication Infrastructure For Standard-Based Teleconferencing Systems", PhD thesis, Technical University Berlin, 1997

[Ott/Perkins] J, Perkins C. june 1999 Internet draft: "Requirements for Local Conference Control ", http://search.ietf.org/internet-drafts/draft-ott-mmusic-mbus-req-00.txt

[Ott J (a)] , Bormann C. "Simple Conference Control Protocol - SCCP", Internet draft 1996

[Ott J (b)], Bormann C (1997) "MTP/SO - Self Organising Multicast" Internet draft 1997, draft-bormann-mtp-so-01.txt

[Pan98]Pan P., Schulzrinne H. "DIAMETER: policy and Accounting Extension for SIP" Internet draft, Internet Engineering task Force, July 1998

[Paxson] V., "End-to-End Routing Behaviour in the Internet", IEEE/ACM Transactions on Networking, Vol.5, No.5, pp. 601-615, October 1997

[Perlman] R., "Folklore of Protocol Design", draft-iab-perlman-folklore-00.txt, Jan 1998, http://sunsite.ics.forth.gr/sunsite/pub/internet-drafts/draft-iab-perlman-folklore-01.txt

[Perry] M., "CONFCNTLR: A Video Conference Controller", Masters thesis in San Francisco State University, December 1997

[Q.1211] ITU Recommendation Q.1211, "Introduction to Intelligent Network Capability Set 1". Technical report, International Telecommunication Union, March 1993.

[Raman] S., McCanne S., "A Model, Analysis, and Protocol Framework for Soft State-based Communication", ACM SIGCOMM technical conference 31 Aug – 2nd Sept, 1999, Cambridge MA

[Rizzo97] Rizzo L., Vicisano L., "A Reliable Multicast Data Distribution Protocol based on Software FEC Techniques", 4th IEEE workshop on the architecture and implementation of high performance communication systems (HPCS'97)

[Rodden] T., Blair G., "CSCW and Distributed Systems: the problem of control", in ECSCW91: The 2nd European conference on CSCW, pp 49-61, Amsterdam, Sept 1991

[Rosenberg] J., Schulzrinne H, "Timer Reconsideration for Enhanced RTP scalability" - Internet draft - draft-ietf-avt-reconsider-00.ps , 1998

[RSVP] RFC 2205, Bradan, Zhang, Berson, "Resource Reservation Protocol", ftp://ftp.isi.edu/in-notes/rfc2205.txt

[RSVP2] Internet draft, Wu L., Davie B., Davari S., "A Framework for Use of RSVP with Diff-serv Networks" http://search.ietf.org/internet-drafts/draft-ietf-diffserv-rsvp-01.txt

[RTP] RFC 1889, Schulzrinne H., Casner S., Jacobson V. "RTP: A Transport Protocol for Real Time Applications" , Internet Engineering Task Force, 1996

[Rubens98] Rubens A., Calhoun P., "DIAMETER Base Protocol" Internet draft, Internet Engineering task Force July 1998

[Sanjoy] Lin, John C., "RMTP: A Reliable Multicast Transport Protocol", IEEE INFOCOM '96, March 1996, pp. 1414-1424. 1996

[Sasse] A., Handley M., Ismail I " Coping with Complexity and Interference: Design Issues in Multimedia Conferencing Systems", Chapter 9 Design issues in CSCW, Duska Rosenberg and Chris Hutchinson, Springer-verlag

[Schooler91] E.M, "Case Study: Multimedia Conference Control in a Packet-switched Teleconferencing System",Journal of Internetworking: Research and Experience, Vol.4, No.2, pp.99-120 (June 1993)

[Schooler92] E.,Stephen L. Casner, "An Architecture for Multimedia Connection Management," in Proc. of 4th IEEE ComSoc International Workshop on Multimedia Communications, (Monterey, California), p. 5, Apr. 1992. also as ISI reprint ISI/RS-92-294.

[Schooler93], "Multiparty Multimedia Session Control (MMUSIC) working group meeting report", in proceedings of the 27th Internet Engineering task force, pp 419-430, 27th IETF meeting in Amsterdam, Amsterdam 1993

[Schulzrinne / Rosenberg NOSSDAV], "A Comparison of SIP and H.323 for Internet Telephony", proceedings of the 1998 Network operating System Support for Digital Audio and Video (NOSSDAV '98), July 1998, Cambridge, England.

[Schulzrinne/ Rosenberg INFOCOM] "Timer Reconsideration for Enhanced RTP Scalability", Proceedings of IEEE Infocom 1998, 27th March – 2nd april, 1998 San Francisco, USA

[SDP] RFC 2327, Handley M., Jacobson V., "Session Description Protocol", ISI/LBNL , IETF , April 1998

[SGCP] Huitema C., Arango M., "Simple Gateway Control Protocol", Bellocore, Internet draft, draft-huitema-sgcp-v1-02.txt, July 1998

[Shenker96] Shenker., Clark d., Estrin D., Herzog S. " Pricing in Computer Networks",*ACM Computer Communication Review*, vol. 26, pp. 19-43, Apr. 1996.

[SIP] RFC 2543, Handley M., Schulzrinne H., Rosenberg J., "Session Initiation Protocol", March 1999

[Speakman98] T., Farincci D., Lin S.(1998) PGM specification - Internet draft draft-speakman-pgm-spec-00.txt, July 1998

[Stevens (a)] R.,"Unix network Programming", Prentice Hall, Vol 1 ISBN 0-13-949876-1, 1990

[Stevens (b)] R. "Advanced Programming in the Unix environment", Prentice Hall, Volume 2, 1992

[Stiller98] Stiller B., Fankhauser G. "Charging and Accounting for Integrated Internet Services – state of the art, problems and trends", INET 1998, Switzerland, July 21 – 24, 1998

[Sudan] M., R. Yavatkar, J. Griffeon, "A Reliable Dissemination Protocol for Interactive Collaborative Applications", ACM multimedia Nov 5 - Nov 9 1995, pp 333 – 344, San Francisco, CA

[T.120] ITU draft recommendation T.120, "Data Protocols for Multimedia Conferencing", 1997

[Varadhan] K., Estrin, D., Floyd S., "Impact of Network Dynamics on End-to-End Protocols: Case Studies in TCP and Reliable Multicast", IEEE INFOCOM, 29march – 2nd april 1998, San Francisco, CA.

[Vicisano] L., Crowcroft J., Rizzo L. "TCP Like Congestion Control for Layered Multicast Data Transfer "- Proceeding of INFOCOM, 29th March – 2nd april, 1998, San francisco, USA

[Vicisano97] L., Rizzo L "A Reliable Multicast Data distribution Protocol based on Software FEC techniques" Proceedings of the 4th IEEE workshop on the architecture and Implementation of High Performance Communication systems (HPCS 1997)

[Wang97] Wang Z., Internet draft "User-Share Differentiation (USD) Scalable Bandwidth Allocation for Differentiated Services", 1997

[Wenger95] Wenger S., Internet draft "RTP Payload and Encoding Schemes for H.263" draft-wenger-rtp-h263plus-payload-00.txt, July 1995

[White98] White P., Crowcroft J., "A Dynamic Sender-Initiated Reservation Protocol for the Internet", 8th IFIP Conference on High Performance Networking (HPN'98) The Millennium Push of Internet, Vienna September 21-25, 1998

[X.Rex] Xu, Zhang H, Yavatkar R., "Resilient Multicast Support for Continuous media applications" "http://research.ivv.nasa.gov/RMP/links.html", in Proc. International Workshop on Network and Operating System Support for Digital Audio and Video(NOSSDAV), St. Louis, May 1997

[Zhang93] Zhang L., Deering S., Shenker S., "RSVP: A New Resource Reservation Protocol", IEEE Network, September 1993

[Zheng]. W, Crowcroft J, Diot C. and Ghosh A "Framework For Reliable Multicast Application Design", HIPPARCH 1997 workshop, Uppsala, Sweden, June 12-13, 1997.

Appendix A

This appendix is divided into two main parts: the first part is an analysis of CCCS, based on formal description technique using language Z, and the second part is on limitation of number of participants and messages that can be generated in a conference. The first part verifies the completeness of this service protocol using formal methods and design specifications. The other part is based on mathematical analysis. Mathematical analysis and formal methods are at their best when applied early in the design stage to capture the system behaviour and enable predictions on typical and boundary condition cases.

Please note that this is a partial specification and analysis to support the works presented in this thesis already, especially in Chapter 3. The formal methods presented in this appendix complements other design approaches that was utilised earlier, such as State transition diagram and well known software engineering Broker design pattern. A system's design work that is solely based on formal description technique will be presented with a full formal specification, whereas this work's design has been mainly influenced by software design patterns for distributed systems that also took the advantage of formal description techniques.

Part A:

High level specification:

CCCS provides a set of conference control services which works in a Broker pattern. So if a client requests for some services from a CCCS broker, it has to response accordingly. This sort of service protocol has to follow certain design rules for it to be a well designed, complete protocol. The folklore of protocol design [Perlman] has been used as a checklist for the protocol's features:

Protocol design criteria	CCCS
Error control	Yes
Well defined problem	Yes – how communication using different conference control mechanisms can be seamlessly integrated into a single mechanism, and to propose a set of Common Conference Control Services

	(CCCS)
Identification	Yes – ConfID
Optimise for most common use	Yes – interoperability and most common activities in a conference are implemented
Large enough fields in the message format	Yes
Reserved field	Yes
Independent of layers	No – dependant on network for reliability
Version number field	Yes
Option field	No

Table 10: Design issues that is fulfilled by CCCS as a service protocol

A taxonomy consisting of useful parameters for describing the session control requirements of large scale multicast applications has been used[Bagnall] to produce a requirement statement for CCCS. A session is normally a gathering consisting of flows of information related by a common description that persists for a non-trivial (more than a few seconds) such that the participants (humans or applications) are involved and interested at intermediate times. A session may be defined recursively as a super or sub set of other sessions.

Session control requirement	Methods and definitions	CCCS
Initiation	Announcement Invitation Directive – specific participants are forced to join	No (provided by clients, e.g SDR) Yes No
Active time	Example: Duration over 30 seconds	Yes
Atomic join	Session fails unless a certain proportion of potential participants accept an invitation	No
Late join allowed		Yes
Temporary leaving allowed		Yes
Late join with catch up	Example: a football game where scores can be requested	No
Available bandwidth for multiple streams per session	Example: H.323 gatekeeper	No
Number of receivers and	Checks for maximum	Yes – only checks up to 10

senders	number	participants now
Security		No
Priority queuing	Example : 999 calls	No – depends on network devices (e.g. routers)
Membership criteria		Yes – hosts, chair and users allowed
Payment and charging		Yes

Table 11: Session control requirement for large scale multicast application vs CCCS

From above it can be seen that CCCS does not meet all the design criteria or the session control requirements. However, the first principle for designing a usable protocol /application is simplicity[Perlman]. Making a protocol or an application flexible enough to fit every possible situation or always finding the theoretically optimal solution, creates a more complex protocol. Therefore, CCCS has been implemented as a simple service protocol which allows for future enhancements such as security and other values added functions if desired.

Table 10 and 11 represent verification of high level design specification and requirements of Common Conference Control Services. The section below concentrates on more specific details of unification of different control activities using the formal description language Z.

CCCS is a canonical model of conference control, i.e. it consists of all common control functions and states that are present in any conferencing. Therefore, all conferences can interoperate using CCCS. In order to unify, CCCS has to check all the matching operations and the operations consistency.

We are trying to do $R \subseteq S \times T$, but operations of S have to be = operations of T.

In the general case, let us assume we have been given state schemas:

```
D1 ______________        D2 ______________
| x:S                    | x: T
| _____                  | _____
| preds                  | predt
|______________          |______________
```

D_1: $x:S$; $pred_s$ $\qquad$ D_2: $x: T$; $pred_t$

The unification of D_1 and D_2 should be:

D————————

$x: S \cup T$

$x \in S \Rightarrow pred_s$

$x \in T \Rightarrow pred_t$

————————

$S \cup T$ is an error unless S and T have the same (maximal) type of operations. Let us take operations of S which could be a SIP based conferencing architecture:

Status1 :: idle | register | invite | trying | ringing | ok | ack | conversation | bye | ok

Now, consider T which is a H.323 based conferencing:

Status2 :: idle | register | call signalling (H.225) | H.225 alerting | H.225 connect | negotiate (H.245) |media exchange (open logical channels) | connect | conversation | close channel | ok

We can form the union of these types by taking the correspondence relation to be {(idle, idle), (register, register), (invite, H.225 invite), (trying, H.225 alerting), (ringing, H.225 connect), ($\perp_1$, H.245 negotiate), ($\perp_2$, H.245 logical channel exchange), (ok, connect), (conversation, conversation), (bye, close channel)} .....(1)

If the correspondence relation is $R \subseteq S \times T$, the inhabitants of the unified state schema will be the tuples of tot R. To account for the predicate $pred_s$, we include a predicate that should hold whenever the $S_\perp$ value is not $\perp_s$, and similarly for $pred_T$:

D (unified state by correspondence)

$x_1: S_\perp; x_2: T_\perp$

————————

$(x_1, x_2) \in \text{tot } R$ .....(2)

$\forall x{:}S \bullet x_1 = \text{just } S\ x \Rightarrow pred_s$

$\forall x{:}T \bullet x_2 = \text{just} T\ x \Rightarrow pred_T$

Using equation (2) and examples from (1) we get:

D

x_1: Status1$_\perp$; x_2 Status2$_\perp$

(x_1 = justStatus1 register $\wedge$ x_2 = justStatus2 register) $\vee$
(x_1 = justStatus1 invite $\wedge$ x_2 = justStatus2 H.225 invite) $\vee$
(x_1 = justStatus1 ok $\wedge$ x_2 = justStatus2 connect)

The implementation of CCCS described in chapter 3 follows exactly what is represented in the above formal specification.

Floor control is a service provided by CCCS, which is not a common service shown in the above equations. By this mechanism one person can send data in one channel to a CCCS server and the server distributes it to all the others. A conference's floor scheme is made up of three main facets which can be applied to any model of conferencing:

1. The FLOOR schema which is the set of channels open at a given time, and previously requested by one of the current people in the conference.
2. The QUEUE schema which maintains an ordered list of Bids, which are made up of the identity of the user, the current users, a desired floor (by default a floor where requester has a communication channel with all people of the conference) and the time of the request. It also has a weighting function which ranks the bids.
3. ALLOWED schema, which restricts the type of floors allowed in a given schema.

So then our total schema for the CONFERENCE is:

CONFERENCE

FLOOR
QUEUE
ALLOWED

Some predicate

- FLOOR schema

Given the set of all people **P**, a floor is a set of ordered pairs of people, where each ordered pair suggests a directional communication channel from the first of the pair to the second. The floor should only be made up of the users currently in the conference, and we keep track of the users of the system. All these information is captured by

FLOOR

$\text{Floor}: \mathbf{P}\,(P \times P)$

$\text{Users}: \mathbf{P}\,P$

$\text{Floor} \subseteq \text{users} \times \text{users}$

Some types of floor is defined below by way of example: the OPENFLOOR is where everybody has a communication channel with everyone else – except themselves.

OPENFLOOR

FLOOR

$\text{Floor} = \{\, x,y{:}P \mid x,y \in \text{users} \bullet (x,y) \,\} - \{\, x{:}P \mid x \in \text{users} \bullet (x,x) \,\}$

Then there is the most common mode of floor where one speaker talks to all the other people in the conference:

SPEAKERFLOOR

FLOOR

$\forall\, p,q{:}P \mid p.q \in \text{floor} \bullet \text{fst}\ p = \text{fst}\ q$

$\#\text{floor} = \#\text{users} - 1$

Let us define a function SPEAKER which takes a set of users and a floor and return the number of users to which the SPEAKER is talking to, that is, the number of channels from the speaker to other users:

SPEAKER?

$\text{Speaker}: \mathrm{P}\,(P \times P) \rightarrow P \rightarrow \mathrm{N}$

$\forall F: \mathrm{P}\,(P \times P),\ u{:}\mathrm{P}$

$\text{Speaker}\ f\ u = n$

$\Leftrightarrow$

$\#\{\, p{:}P \mid (u,p) \in f \bullet p \,\} = n$

There are two cases for a floor f, and a user u, either

1. speaker f u = 0

In which case p is the only listener of the current floor

2. speaker f u > 0

In which case p is talking to someone in that floor. The number of users who are listening to the speaker p can be limited by factors like bandwidth and system description. This will be discussed in part B.

- Operations on the FLOOR schema

There are certain operations that take place on the FLOOR schema, including add user, remove user, speak and listen. When a new user joins the conference system, he or she by default will be added to the listeners of any maximal speaker (where maximal speaker is defined a default speaker who speaks to users in the conference more than anyone else).

ADDUSER ———————————

$u? : P$

$\Delta FLOOR$

$u? \notin users$

$users' = users \cup \{u?\}$

$floor' = floor \cup \{\ q{:}\mathit{P} \mid (q \text{ maximal floor}) \bullet (q,u?)\}$

(Note: in all examples u? represents the person initiating the operation). The user may not be very happy about listening to maximal speaker p? by default. So if a user u? wishes to stop hearing the speaker p?:

REMOVE-LISTEN-SPEAKER———————

$u?, p?{:}\mathit{P}$

$\Delta\ FLOOR$

$u?, p? \in users$

$(p?, u?) \in floor$

$floor' = floor - \{(p?, u?)\}$

$users' = users$

Correspondingly, any user should be able to get to hear any speaker of a floor: (this operation is optional, it may not be allowed).

REQUEST-LISTEN-SPEAKER ————

$u?, p?{:}P$

FLOOR

$u?, p? \in users$

$speaker\ floor\ p? > 0$

$floor' = floor\ U\ \{(p?, u?)\}$

$users' = users$

The first predicate in the above schema states that p? is already speaking – there is not much point listening to them if they are not. If the speaker was not specified, then the default would be for a channel to be set up from the maximal speaker of the current floor to the user u?.

When a user leaves the conference he must not be a speaker of the current floor. If that is true, then any open channels involving the user should be closed and the user needs to be removed from the variable users.

```
LEAVE – P ──────────
  u? : P
  Δ FLOOR
──────────────────────
Speaker u? floor = 0
floor’ = floor – {p:P | (p, u?) floor • (p, u?)}
users’ = users – {u?}
```

When the conference is finished and there are no more speakers or listeners left, close all the channels.

```
FINISHED ──────────
u?:P
Δ FLOOR
──────────────────────
speaker floor u? = 0
floor’ = floor – {p:P | (u?,p) ∈ floor • (u?, p)}
users’ = user
```

- The QUEUE SCHEMA

This consists of an ordered sequence of ranked “bids” for desired floors. A Bid is defined as follows:

```
BID ──────────────────
FLOOR
requester: P
id : N
Time of request : Time
S: SPEECH – ACT
──────────────────
Request ∈ FLOOR.users
```

There are two main operations that affect the QUEUE schema, requesting a bid and removing a bid.

```
NEW -BID ———
B?:Bid
QUEUE - BIDS
Ran queue' = ran queue U {b?}
```

For removing a bid, the precondition is that the bid was actually made by the person removing the bid:

```
REMOVE -BID———
u? : P
id: N
ΔQUEUE - BIDS
————————————————————
∃ b:Bid | b ∈ ran queue b.id – id
b.requester = u?
ran queue' = ran queue – {b}
```

- The ALLOWED schema

This schema just gives the set of allowed floors.

```
ALLOWED
Allowed:PP(P x P)
```

The conference schema can now be completed using all the operation discussed above which is how CCCS implements its floor control operation:

```
 CONFERENCE——————
FLOOR
QUEUE - BIDS
ALLOWED
——————————
Floor  allowed
∀ F: Bid | f ∈ ran queue • f.floor ∈ allowed
```

The next section is a mathematical analysis on limitation of number of participants:

Part B

Limits on participants in a Group:

Given that R_{pkt} is the rate of packet generation, S_{pkt} is the size of each packet, and B_{net} is the network bandwidth, the limit on the maximum number of media sources N_{max}, that can simultaneously transmit packets onto the network, is given by:

$$N_{max} * R_{pkt} * S_{pkt} <= B_{net}$$

$$\Rightarrow N_{max} <= \frac{B_{net}}{R_{pkt} * S_{pkt}} \qquad ...(1)$$

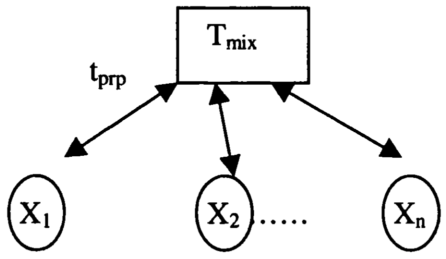


Figure A.1: Logical diagram of number of participants in a group

Note that N_{max} provides an upper bound on the number of participants that can be supported in a conference. If the participants of a conference span multiple networks (as opposed to sharing bandwidth on the same network, as assumed in this analysis), the value of N_{max} can be increased even further. The bound on the maximum number of participants (Np) depends on the limitation in the rate of packet reception at the network-host interface, and the packet processing overhead. The following analysis derives a bound on the value of Np:

During each packet generation period, Np packets are sequentially received on the network by a mixer. Since the propagation delay for each packet is t_{prp}, the total round trip delay for a certain node is $2t_{prp}$ as shown in Figure A.1. Assuming that mixing is performed by averaging of media data (as in the case of audio), the maximum time for mixing Np media packets is (Np –1) * t_{mix}, where t_{mix} is the time to mix two media packets (sec). In the presence of multicasting, the mixer has to perform only one transmission to send the mixed media packet to all the participants. Since all the above functions must be performed during each packet generation period, and if it is assumed that the acceptable delay for voice from speaker to listener is $T_{pkt} = 400$ millisecond, we obtain:

$$(Np - 1) * t_{mix} + 2t_{prp} <= T_{pkt}$$
$$\Rightarrow Np <= \frac{T_{pkt} - 2t_{prp}}{T_{mix}} + 1 \qquad \ldots(2)$$

It can be observed from equation 1 that the number of participants in a group is constrained by the network bandwidth. From equation 2, it is also observed that it is dependant on the mixer and the propagation delay. If a large number of participants cause increase in mixing delay, it results in a poor quality conference. It is also derived that the smaller the delay for round trip, the number of participants in a group gets larger.

Appendix B

This appendix is an extension for Cchapter 3, section 3.3.5. Here, the messages generated from SIP which is a text based protocol are listed and the function mapping are shown:
(Please note: the text below mainly presents debugging messages generated by SIP capable terminals, whereas H.323 ASN.1 decoded cannot be presented here without getting prior copyright permission from the H.323 vendor)

1. The Network Configuration

```
       +--------+                      +--------+
       |        |                      |        |
   +---|  SIP#1 |<--------SIP--------->|  SIP#3 |---+
   |   |        |                      |        |   |
   |   +--------+                      +--------+   |
   |                                                |
 |----|                                          |-----|
 CCCS                                            |CCCS |
 |----|                                          |-----|
   |                                                |
   |   +----------------------------------------+   |
   +---|                H.323                   |---+
       +----------------------------------------+
```

2. Test Cases

The following scenarios should be used to test interoperability:

a) Basic Call Flow (including set-up and tear-down)

b) Busy (call made to a busy number)

c) Ring-No-Answer

The traces and call-flows for each of these are presented in the next section.

3. Call-flows and Traces

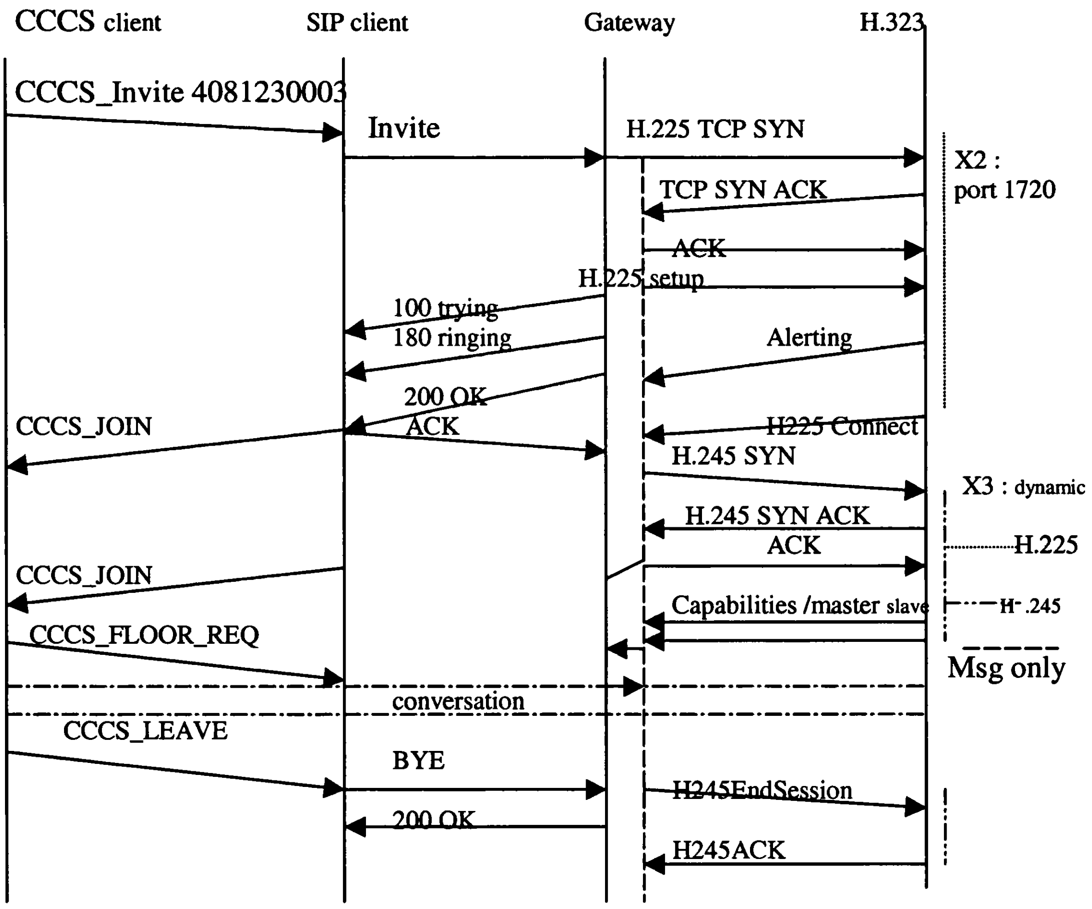
In the call-flows shown below, the SIP messages between the Mbone terminals have been numbered. These numbers correspond to the appropriate SIP messages (following the call-flow).

Note: In the traces there is a period at the end of each line. This indicates a new-line (the sniffer prints a period for a new-line).

a) Basic Call Flow:

Messages 1 to 7 trace successful connected end-to-end call.

```
1: INVITE sip:4081230003@everest SIP/2.0.
   Via: SIP/2.0/UDP fujiyama.
   From: 4081230001@fujiyama.
   To: 4081230003@everest.
   Call-ID: LEV55519990720180211264011@fujiyama.
   CSeq: 1 INVITE.
   Content-Type: Application/sdp.
   Content-Length: 123..
   v=0.
   o=Level3 2580642516 2580642516 IN IP4 gibralter.
   s=VoIP Call.
   c=IN IP4 192.168.0.94.
   t=2580642516 0.
   m=audio 3400 RTP/AVP 0.
```



```
2: SIP/2.0 100 TRYING.
   Via: SIP/2.0/UDP fujiyama.
   From: 4081230001@fujiyama.
```

```
    To: 4081230003@everest.
    Call-ID: LEV5551999072018021126401 1@fujiyama.
    CSeq: 1 INVITE.
    Content-Type: Application/sdp.
    Content-Length: 123..
    v=0.
    o=Level3 2580642516 2580642516 IN IP4 gibralter.
    s=VoIP Call.
    c=IN IP4 192.168.0.95.
    t=2580642516 0.
    m=audio 3400 RTP/AVP 0.

 3:  SIP/2.0 180 RINGING.
    Via: SIP/2.0/UDP fujiyama.
    From: 4081230001@fujiyama.
    To: 4081230003@everest.
    Call-ID: LEV5551999072018021126401 1@fujiyama.
    CSeq: 1 INVITE.
    Content-Type: Application/sdp.
    Content-Length: 123..
    v=0.
    o=Level3 2580642516 2580642516 IN IP4 gibralter.
    s=VoIP Call.
    c=IN IP4 192.168.0.95.
    t=2580642516 0.
    m=audio 3400 RTP/AVP 0.


 4:  SIP/2.0 200 OK.
    Via: SIP/2.0/UDP fujiyama.
    From: 4081230001@fujiyama.
    To: 4081230003@everest.
    Call-ID: LEV5551999072018021126401 1@fujiyama.
    CSeq: 1 INVITE..

 5:  ACK sip:4081230003@everest SIP/2.0.
    Via: SIP/2.0/UDP fujiyama.
    From: 4081230001@fujiyama.
    To: 4081230003@everest.
    Call-ID: LEV5551999072018021126401 1@fujiyama.
    CSeq: 1 INVITE..

 6:  BYE sip:4081230003@everest SIP/2.0.
    Via: SIP/2.0/UDP fujiyama.
    From: 4081230001@fujiyama.
    To: 4081230003@everest.
    Call-ID: LEV5551999072018021126401 1@fujiyama.
```

CSeq: 2 BYE..

7: SIP/2.0 200 OK.
Via: SIP/2.0/UDP fujiyama.
From: 4081230001@fujiyama.
To: 4081230003@everest.
Call-ID: LEV55519990720180211264011@fujiyama.
CSeq: 2 BYE..

b) Busy
Messages 1 through 4 trace a call made to a busy number.

```
Cccs           SIP#1              H.323#1
  |              |                 |<---h.225----|
 1|              |<-----INVITE-----|             |
  |<-cccs_inv---|                  |             |
 2|              |----100 TRYING-->|             |
  |-----cccs--->|                  |             |
 3|              |----486 BUSY---->|             |
  |<----cccs----|                  |  dis        |
  |---------→   |                  |--connect--->|
 4|              |<-----ACK--------|             |
  |              |                 |<----ACK-----|
```

1: INVITE sip:4081230001@fujiyama SIP/2.0.
Via: SIP/2.0/UDP everest.
From: 4081230003@everest.
To: 4081230001@fujiyama.
Call-ID: LEV55519990720180944263010@everest.
CSeq: 1 INVITE.
Content-Type: Application/sdp.
Content-Length: 123..
v=0.
o=Level3 2580642516 2580642516 IN IP4 gibralter.
s=VoIP Call.
c=IN IP4 192.168.0.95.
t=2580642516 0.
m=audio 3400 RTP/AVP 0.

2: SIP/2.0 100 TRYING.
Via: SIP/2.0/UDP everest.
From: 4081230003@everest.
To: 4081230001@fujiyama.
Call-ID: LEV55519990720180944263010@everest.
CSeq: 1 INVITE.
Content-Type: Application/sdp.
Content-Length: 123..
v=0.

o=Level3 2580642516 2580642516 IN IP4 gibralter.
s=VoIP Call.
c=IN IP4 192.168.0.94.
t=2580642516 0.
m=audio 3400 RTP/AVP 0.

3: SIP/2.0 486 Busy Here
Via: SIP/2.0/UDP everest.
From: 4081230003@everest.
To: 4081230001@fujiyama.
Call-ID: LEV555199907201809442630l0@everest.
CSeq: 1 INVITE.

The SIP#1 would receive a cccs_error (with cause code 'User Busy) from its CCCS-side. SIP#1 should send a 486 – 'Busy Here' message to H.323#1 (via GS). This means that the end-system was contacted successfully, but the callee is currently not willing or able to take additional calls. The response may indicate a better time to call in the Retry-After header.

4: ACK sip:4081230001@fujiyama SIP/2.0.
Via: SIP/2.0/UDP everest.
From: 4081230003@everest.
To: 4081230001@fujiyama.
Call-ID: LEV555199907201809442630l0@everest.
CSeq: 1 ACK..

SIP#1 should then send an ACK to CCCS to confirm receipt of this final response.

c) Ring-No-Answer
Messages 1-5 trace Ring-No-Answer.

```
CCCS           SIP#1              H.323#1
  |----MSG1---->|                  |             |
 1|             |-----INVITE----->|             |
  |             |                  |----h.225--->|
 2|             |<---100 TRYING---|             |
  |             |                  |<----CONN----|
 3|             |<--180 RINGING---|             |
  |<----MSG2----|                  |             |
  |             |                  |             |
  |-----ERR---->|                  |             |
  |             |                  |             |
 4|             |------BYE------->|             |
 5|             |<----200 OK------|             |
  |             |                  |-----ERR---->|
  |             |                  |<----ACK-----|
  |             |                  |             |
  |             |                  |             |
```

1: INVITE sip:4081230003@everest SIP/2.0.

```
   Via: SIP/2.0/UDP fujiyama.
   From: 4081230001@fujiyama.
   To: 4081230003@everest.
   Call-ID: LEV5551999072018021126401l@fujiyama.
   CSeq: 1 INVITE.
   Content-Type: Application/sdp.
   Content-Length: 123..
   v=0.
   o=Level3 2580642516 2580642516 IN IP4 gibralter.
   s=VoIP Call.
   c=IN IP4 192.168.0.94.
   t=2580642516 0.
   m=audio 3400 RTP/AVP 0.

2: SIP/2.0 100 TRYING.
   Via: SIP/2.0/UDP fujiyama.
   From: 4081230001@fujiyama.
   To: 4081230003@everest.
   Call-ID: LEV5551999072018021126401l@fujiyama.
   CSeq: 1 INVITE.
   Content-Type: Application/sdp.
   Content-Length: 123..
   v=0.
   o=Level3 2580642516 2580642516 IN IP4 gibralter.
   s=VoIP Call.
   c=IN IP4 192.168.0.95.
   t=2580642516 0.
   m=audio 3400 RTP/AVP 0.

3: SIP/2.0 180 RINGING.
   Via: SIP/2.0/UDP fujiyama.
   From: 4081230001@fujiyama.
   To: 4081230003@everest.
   Call-ID: LEV5551999072018021126401l@fujiyama.
   CSeq: 1 INVITE.
   Content-Type: Application/sdp.
   Content-Length: 123..
   v=0.
   o=Level3 2580642516 2580642516 IN IP4 gibralter.
   s=VoIP Call.
   c=IN IP4 192.168.0.95.
   t=2580642516 0.
   m=audio 3400 RTP/AVP 0.

4: BYE sip:4081230003@everest SIP/2.0.
   Via: SIP/2.0/UDP fujiyama.
```

```
     From: 4081230001@fujiyama.
     To: 4081230003@everest.
     Call-ID: LEV5551999072018021126401l@fujiyama.
     CSeq: 2 BYE..

5:  SIP/2.0 200 OK.
     Via: SIP/2.0/UDP fujiyama.
     From: 4081230001@fujiyama.
     To: 4081230003@everest.
     Call-ID: LEV5551999072018021126401l@fujiyama.
     CSeq: 2 BYE..
```

Appendix C

This appendix is an extension of Chapter 3, Section 3.8. Here, all the IN operations that are applicable for this thesis are listed.

- *Abbreviated calling (ABD)*

Abbreviated dialing allows the definition of short (e.g. two digit) digit sequences to represent the actual dialing sequence for a public or private numbering schemes.

This translation could be performed by having end systems configured to consult a local database server (running e.g. LDAP) for address-translation queries.

This translation could also be performed by a local proxy or redirection server which the end system always sends it outgoing call requests. This facility can be used by both H.323 and SIP. CCCS's internal management feature GS, will be enhanced from this service.

- *Call distribution (CD)*

This service feature allows the served user to specify the percentage of calls to be distributed among two or more destinations. The proxy server of SIP or gatekeeper for H.323 can make this decision. However, none of these two protocols currently provide this functionality.

- *Call forwarding on busy/don't answer (CFC)*

This service feature allows the called user to forward particular calls if the called user is busy or does not answer within a specified number of rings. This is not provided in H.323 or SIP at the moment.

- *Call gapping (GAP)*

This feature allows the service provider to restrict the number of calls to a served user to prevent congestion of the network. This feature is not provided in H.323. SIP with a policy server may handle this scenario. However, because of the nature of the

Internet, it is not very clear how SIP will be discarding calls like SS7 does on the telephone network. It also shows that SIP policy server requires advanced security features to stop users bypassing the server.

- *Call hold with announcement (CHA)*

The call hold with announcement service feature allows a subscriber to place a call on hold with options to play music or customized announcements to the held party. H.323 is in the process of standardizing H.450 (4) this feature. This standard is contributed from ITU sg-16. SIP does not provide this feature, however RTSP is capable of providing this feature.

- *Call logging (LOG)*

This service feature allows for a record to be prepared each time that a call is received to a specified telephone number. H.323 or SIP does not provides this feature, but the IP addresses of the source can be monitored which may not be completely reliable.

- *Call queuing (QUE)*

This service feature allows calls which would otherwise be declared busy to be placed in a queue and connected as soon as the free condition is detected. Neither H.323 or SIP provides this feature at the moment.

- *Call transfer (TRA)*

The call transfer service allows a subscriber to place a call on hold and transfer the call
to another location. ITU sg16 has standardised the call transfer supplementary services for H.323 H.450(2). The SIP call control draft is also considering these decentralized call transfer services.

- *Call waiting (CW)*

This service feature allows a subscriber to receive a notification that another party is trying to reach his number while he is busy speaking to another person. H.323 has standard H.450(6) to handle this feature while SIP does not provide this feature yet.

- *Consultation calling (COC)*

The consultation calling service feature allows a subscriber to place a call on hold, in order to initiate a new call for consultation. H.323 has H.450(8) for this feature, while SIP does not provide this feature.

- *Customer profile management (CPM)*

This service feature allows the subscriber to real-time manage his service profile, i.e. terminating destinations, announcement to be played etc. This feature can be implemented in any end system, and it is also known as call management which can be provided in SIP and H.323v2.

- *Customer recorded announcement (CRA)*

This service allows a call to be completed to a (customized) terminating announcement instead of a subscriber line. The served user may define different announcements for unsuccessful call completions due to different reasons. Although RTSP provides this feature, neither SIP or H.323 provides CRA facilities.

- *Customized ringing (CRG)*

This feature allows the subscriber to allocate a distinctive ringing to a list of calling parties. ITU is in the process of standardising this feature while SIP draft claims that this feature can be provided in end-system without giving further details on it.

- *Follow-me diversion (FMD)*

With this service feature, a user may register for incoming calls to any terminal access. H.323 Gatekeeper and SIP REGISTER message provide this feature very easily.

- *Meet-me conference (MMC)*

This service feature allows the user to reserve a conference resource for making a multi-party call. The nature of H.323 calls allows this feature to be easily implemented by a gatekeeper, while in SIP this requires further investigation.

- *Multi-way calling (MWC)*

This feature allows the user to establish multiple, simultaneous telephone calls with other parties. With SIP this has been the building block from the beginning, therefore with the use of multicast this is easily scalable. Whereas, H.323 uses MCU to provide multi-way calls .

- *One number (ONE)*

This allows for example, businesses to advertise just one telephone number throughput their market area. SDR provides this feature easily and H.323 v3 can provide this feature using LDAP.

- *Originating call screening (OCS)*

This service feature allows the caller to bar calls from certain areas based on the district code of the area from which the call is originated. Since the nature of Internet does not really depend on distance or district code this feature is not really applicable for SIP or H.323. However, similar feature can be implemented in SIP or H.323 using the IP addresses instead of district codes of the area.

- *Premium charging (PRMC)*

This service feature allows for the pay back of part of the cost of a call to the called party. In an Internet environment this service could either be negotiated directly between the two parties, or some external settlement authority trusted by both parties could be used.

- *Private numbering plan (PNP)*

This service feature allows the subscriber to maintain a numbering plan within his private network, which is separate from the public numbering plan. Since in the Internet environment, addressing is always controlled by independently administered systems, this largely becomes trivial. If a numbering plan should be hidden or partially hidden from the public, a proxy can pretend to know nothing about the private addresses when they come from outside.

- *Time dependant routing (TDR)*

This service feature allows the served user to apply different call treatments based on the time of day, day of week, holiday etc. This can be easily provided by SIP and H.323