A QoS Content Adaptation Framework for Nomadic Users

by

Khalil Mehdi El-Khatib

A thesis submitted to the Faculty of Graduate and Post-Doctoral Studies in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering

Ottawa-Carleton Institute for Electrical Engineering
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

Abstract

The tremendous growth of the Internet has introduced a number of interoperability problems for distributed multimedia applications. These problems are related to the heterogeneity of client devices, network connectivity, content formats, and user's preferences. The diversity of client devices posed some challenges in aligning and customizing the exchanged data between different users using different devices and with different preferences. Another trend in the telecommunication world that is starting to surface is ubiquitous computing environment where there is a shift of computing technology from the desktop to the background. One of its most notable attributes is its potential to extend the scope of the user's reachability.

The purpose of this thesis is to present a framework for multimedia content adaptation that addresses diversity, heterogeneity, and ubiquity. The framework takes into consideration the profile of communicating users, devices, network connectivity, exchanged content format, context description, and available adaptation services to find a chain of adaptation services that could be applied to the content. Major part of the framework is a QoS-based selection algorithm that finds the best sequence of adaptation services and their QoS configuration that can maximize the user's satisfaction with the delivered content. A restricted version of the framework is used to find the best combination for the number of media streams and their corresponding configuration for presentational multimedia applications.

The framework also forms the core of an architecture for supporting personal mobility. In an environment where a user has access to many communication devices, it is more convenient that the system makes the choice between all these devices depending on previously defined user preferences. The system will also select the values for the QoS parameters that give the user the best satisfaction with the session. We will also describe an extension of the architecture to support service and personal mobility in an ubiquitous computing environment.

Protecting the user's privacy is also addressed in this thesis, with a focus on protecting the confidentiality of the user's preferences during session negotiation.

Acknowledgments

First, I would like to thank my supervisor, Prof. G.v. Bochmann, for his confidence, support, guidance, valuable feedback and continuous encouragement through the course of this work.

I would also like to thank my friends: Rami Dalloul, Gabriel Aghazarian, Khaled El-Fakih, Mouhcine Guennoun, Wissam Mounzer, Nada Tamime, Raja Khalil, Jana Sukarriah, Peter Alleyne, and Nehmeh Baghdadi for their patience and unceasing moral support throughout the course of my studies. I will never ever forget their friendship.

I would also like to thank the following people at NRC: my manager Larry Korba at NRC for his constant support, Ronggong Song for his help and friendship, and Ian Colby for his continuing technical support.

I would like also to thank many member of the Distributed System Research Group (DSRG), especially Eric Zhang, Nabila Hadibi, Yu Zhong, and Bogdan Soloman for providing an interesting and inspiring research environment. Special thanks go to Dr. El-Saddik. I am really grateful to all of them for giving me the chance to work with them.

I would like to thank all the system staff and especially Michel Racine, Keith White, and Roger Montcalm for keeping our machines up and running. Further thanks go to all people working in the administrative office at our school.

Special thanks for Dr. Boukerche for his guidance and support.

I would like also to thank my family for their love, patience, understanding and dealing with me being absent. I am especially grateful to the distinguishable friendship with my sister Nadia; may her soul rest in peace. Additionally, I would like to thank my wife's family for their support.

Last, but by no mean least, I would like to thank my loving wife, Amal Karboubi, for being with me and for encouraging me to fulfill my dream. I promise, I will make it up to you.

Contents

Abstract
Acknowledgment
Contents
List of Figures
List of Tables

1 **Introduction** **11**
1.1 Introduction and motivation 11
1.2 Contribution 12
1.3 Organization 13
2 **Multimedia Applications and Quality of Service (QoS) Management** **17**
2.1 Introduction 17
2.2 Multimedia applications: Definition 17
2.3 Multimedia applications: Constraints and requirements 18
2.3.1 Temporal constraints 18
2.3.2 High data bit rate 19
2.3.3 Service guarantees 19
2.3.4 Group communication 19
2.4 Classes of multimedia applications 20
2.4.1 Presentational multimedia application 20
2.4.2 Conversational multimedia application 21
2.4.3 Messaging multimedia application 21
2.5 Quality of Service: Definition and terms 21
2.5.1 Network level QoS 22
2.5.1.1 Integrated Service 23
2.5.1.2 Resource ReSerVation Protocol 25
2.5.1.3 Differentiated Service 26
2.5.1.4 Multi-Protocol Label Switching 27
2.5.1.5 Constraint-based routing 29
2.5.2 Application level QoS 30
2.5.3 User level QoS 31
2.6 Conclusion 34
3 **Voice-over-IP and Supporting Protocols** **36**
3.1 Introduction 36
3.2 Voice-over IP: What is it? 37
3.3 Benefits of VoIP 37
3.4 Issues with VoIP 38
3.4.1 Lack of QoS support in the Internet 38
3.4.2 Billing 40

3.4.3 Inter-working and transparency of operation 41
3.5 VoIP signaling protocols 41
3.5.1 H.323 41
3.5.1.1 Terminal 42
3.5.1.2 Gatekeeper 42
3.5.1.3 Gateway 43
3.5.1.4 Multipoint control and processing unit 43
3.5.1.5 H.323 protocols 43
3.5.2 Session Initiation Protocol (SIP) 45
3.5.2.1 SIP main entities 46
3.5.2.2 SIP User Agent 46
3.5.2.3 SIP servers 46
3.5.2.4 SIP addressing 47
3.5.2.5 SIP operations models 47
3.5.2.6 SIP messages 48
3.5.3 Comparison of H.323 and SIP 49
3.6 VoIP supporting protocols 51
3.6.1 RTP and RTCP 51
3.6.2 Session Description Protocol 52
3.6.3 Session Announcement Protocol 54
3.6.4 Megaco/H.248 55
3.7 Conclusion 55
4 A QoS-Based Framework for Distributed Content Adaptation 57
4.1 Introduction 57
4.2 Content adaptation 59
4.3 Required elements for content adaptation 61
4.4 QoS selection algorithm 65
4.4.1 User's satisfaction as selection criteria 66
4.4.2 Extending user's satisfaction to support weighted combination and multi-user conference sessions 68
4.4.3 Constructing a directed graph of trans-coders 70
4.4.4 Graph optimization 73
4.4.5 Adding constraints to the edges 75
4.4.6 QoS selection algorithm 75
4.4.7 Example 79
4.5 Discovering intermediary trans-coding service 81
4.6 Performance analysis of the QoS selection algorithm 83
4.7 Conclusion 85
5 Selecting the QoS Parameters for Multicast Applications Based on User Profile and Device Capability 86
5.1 Introduction 86
5.2 Literature review 88
5.3 Selecting QoS parameters for large groups of users 91
5.3.1 Selecting QoS parameters with unlimited throughput in the source 92
5.3.2 Selecting QoS parameters and channels with limited throughput in the source 93

5.3.2.1 Selecting the combination of content delivery channels: Problem definition 95
5.3.2.2 Selecting the group representative ... 96
5.3.2.3 A heuristic algorithm for QoS parameter and channel selection ... 98
5.3.2.4 Performance evaluation of the OCoV heuristic algorithm ... 100
5.4 Conclusion ... 102
6 **Architectures for Personal Mobility Systems ... 104**
6.1 Introduction ... 104
6.2 Personal mobility: Problem statement ... 105
6.3 Architectures for personal mobility ... 108
6.3.1 The Mobile People Architecture (MPA) ... 108
6.3.2 ICEBERG: Internet-core Network Architecture for Integrated Communications ... 111
6.3.3 Telephony Over Packet networkS (TOPS) ... 113
6.3.4 Call Management Agent System ... 116
6.3.5 Session Initiation Protocol ... 118
6.3.6 Seamless Personal Information Networking (SPIN) ... 119
6.3.7 Personal mobility in telecommunication networks. ... 121
6.4 What is still missing in personal mobility? ... 122
7 **MobInTel: Mobile Internet Telecommunication architecture ... 124**
7.1 Introduction ... 124
7.2 Home Directory ... 126
7.2.1 Content of the Home Directory ... 126
7.2.2 Subscription to the HD ... 128
7.2.3 Updating the content of the HD ... 128
7.2.4 Benefits of the HD ... 129
7.3 Architecture of the HDA ... 130
7.4 Automatic device and QoS parameter selection ... 131
7.5 Support for session establishment in MobInTel ... 132
7.6 Extending MobInTel to support personal and service mobility in ubiquitous computing environments ... 134
7.7 Architectures for personal and service mobility in ubiquitous computing environments ... 136
7.8 Extension to the MobInTel architecture ... 138
7.9 Support for service mobility ... 141
7.10 Usage scenario (continued) ... 142
7.11 Experimentation and evaluation ... 143
7.11.1 Hardware, software, and communication protocols ... 144
7.11.2 Experimental environment ... 145
7.11.3 Results ... 147
7.12 Conclusion ... 149
8 **Preserving the Privacy of User Preferences in Multimedia Communication ... 150**
8.1 Introduction ... 150
8.2 Literature review ... 151
8.3 Privacy invasion in multimedia applications ... 153
8.4 Proposed solutions ... 155

8.4.1 Using a trusted third party 155
8.4.2 Using P3P/APPEL to announce privacy policy and build trust 156
8.4.3 Using distributed secure computation 157
8.5 Using oblivious polynomial evaluation to negotiate private preferences 157
8.5.1 Oblivious Transfer 157
8.5.2 Oblivious polynomial evaluation protocols 159
8.5.3 Privacy Preserving Negotiation Protocol 160
8.6 Complexity and efficiency 162
8.7 Conclusion and discussion 164
9 Conclusion and Future Work 166

Figures

Figure 1. A conceptual model of QoS. 22
Figure 2. An MPLS enabled domain []..... 27
Figure 3. Example of the mapping function determining the user satisfaction depending on the value of a QoS parameter..... 32
Figure 4. H.323 environment..... 42
Figure 5. H.323 protocol stack..... 44
Figure 6. RTP header..... 51
Figure 7. Framework for content adaptation..... 66
Figure 8. Possible satisfaction function for the frame rate. 67
Figure 9. Trans-coder with multiple input and output links 71
Figure 10. Directed trans-coding graph 72
Figure 11. Pseudo-code for the graph construction algorithm..... 73
Figure 12. Pseudo-code for the graph optimization..... 74
Figure 13. Optimized directed trans-coding graph 74
Figure 14. Structure of the trans-coding service. 76
Figure 15. Pseudo-code for the route selection algorithm 77
Figure 16. Graph selection..... 78
Figure 17. Example of trans-coding graph 79
Figure 18. Trans-coders along the BGP AS path..... 82
Figure 19. Optimized graph construction algorithm..... 82
Figure 20. Graph creation time 84
Figure 21 User satisfaction using different selection set of trans-coders. 84
Figure 22. Server bandwidth limit vs. average satisfaction..... 94
Figure 23. Server throughput limit vs. number of streams 94
Figure 24. Assigning receivers to different variants..... 95
Figure 25. Average satisfaction with different variants of grouping. 98
Figure 26. The pseudo code of the OCoV heuristic algorithm. 100
Figure 27. Running time for the OCoV heuristic algorithm and the optimal algorithm. 101
Figure 28. Difference in the total satisfaction between the OCoV heuristic algorithm and the optimal algorithm. 102
Figure 29: Architecture of the MPA *personal proxy*. 109
Figure 30: The ICEBERG Architecture (from [136])..... 113
Figure 31: Structure of a user record 114
Figure 32. A CMA manages multiple communication terminals..... 116
Figure 33. Example of personal mobility support in SIP..... 119
Figure 34: SPIN system architecture 120
Figure 35: User Profile..... 127
Figure 36. Components of the HDA..... 131
Figure 37: HDA support for session establishment. 133
Figure 38. HDA support for connecting two PSTN phones. 134
Figure 39. Components of the Personal Agent 140
Figure 40. Session establishment based on the Personal Agent 143
Figure 41. Experiment environment layout 146

Tables

Table 1. Different types of traffic put different QoS demands on networks. 23
Table 2. Label Information Base 29
Table 3. SIP request methods..... 48
Table 4. Response Status-Code and Categories..... 49
Table 5. Results for each step of the path selection algorithm 80
Table 6. Simulated user population. 94
Table 7. Variants of the preferences selection for the group representative..... 97
Table 8. Transcoders used in the prototype 145
Table 9. Session setup time..... 147
Table 10. Message sequence for OT_1^2. 158
Table 11. Chang's OPE..... 160
Table 12. Compressed version of the privacy preserving negotiation protocol 163

Appendices

Appendix A. User Profile Schema…….…….…….…….……...189
Appendix B. Proxy Schema…..……...192
Appendix C. Trans-coder Schema……...196
Appendix D. Acronyms……...…197

Chapter 1

Introduction

1.1 Introduction and motivation

One of the major breakthroughs of the twenty first century is the wide spread of affordable telecommunication network technologies to the majority of people: cellular phone, wired phone, pager, Personal Digital Assistant (PDA) with network connectivity, to list a few. Other new services such as Internet-TV, Internet-phone are also starting to get acceptance. People now use different devices and applications for different roles and different tasks throughout their daily life.

While each device presents an additional avenue to its owner, this multiplicity of communication devices has left users stranded in the middle. For a caller, it became a challenge to find the callee at a convenient time and through the right device, and more challenging for callees who would love to have an easy way to customize how people can reach them according to their schedule and preference while keeping their privacy.

This diversity of client devices also posed some challenges in aligning and customizing the exchanged data between different users with different preferences. The challenge is even bigger for multimedia content providers who are faced with the dilemma of finding the combination of different variants of the content to create, store, and send to their subscribers that maximize their satisfaction and hence keep on coming back.

Another trend in the telecommunication world that is starting to gain popularity is ubiquitous computing, where multiple small devices can team up together to provide innovative services to the occupants of the environment. Combined with advances in short-range wireless communications, a ubiquitous environment presents an environment rich in services for the users. The challenge is that occupants of the environment are usually un-aware of the available devices and services in the ubiquitous environment and

of their characteristics. Making these devices available at the finger tips of the user adds another dimension to the challenge of delivery of multimedia content.

1.2 Contribution

Personalization is an important factor for the development, evolution, and acceptance of Internet multimedia applications. It has the potential to solve the problem of diversity in users' preferences, user devices, and multimedia content. It also increases the user's satisfaction with these services and encourages their usage. The success of any service that attempts to deliver desirable levels of Quality of Service (QoS) for the future Internet multimedia services should be based, not only on the progress of network transport technology, but also on user's requirements. Only by integrating the user's requirements and preferences, content and context description, and available network and trans-coding resources into the service customization can the utility of the future Internet services be maximized.

The main contribution of our work is a framework for including the user's main preferences, his/her device capabilities, content and context description, as well as the network characteristics and available adaptation services in the session management of these multimedia applications. User's preferences are the main factors to decide what media format to use, and which device to use in order to increase the user's overall satisfaction with the session. Based on all these inputs, the session management protocol might also suggest certain intermediate adaptation services to be applied on the content to increase the user's appreciation and satisfaction.

The key objective of the framework is to allow users to experience personalized communication services. The framework is based on the idea of having profiles, when possible, for all session participants, for their devices, their access networks, available adaptation service, their current context, and the exchanged content. The framework uses the user's satisfaction as optimization criteria when determining the configuration parameters for each component of the communication session. The framework integrates work on metadata description with optimization of usability/utility of multimedia applications. The benefits of the framework are two-fold: on one hand, the over-all quality perceived by the user will be higher, and on other hand, Internet content and service providers would benefit from wider acceptance and usage of their services.

Additionally, we also applied the same concept of using just the user profile and device capability to control the delivery multimedia content to large population of receivers. We used the same approach to select the number of streams, as well as their configuration parameters that can help increase the average satisfaction of all the receivers.

The framework can be used as the core of an architecture that bestows personal mobility, which is an important requirement for users who can be reached on several devices. Based on the user profile, device capabilities, available trans-coder, network, content and context profiles, the architecture selects transparently the communication device as well as the QoS parameters on the device that maximize the user's satisfaction with the communication session. The architecture relieves the user from the task of selecting the device to use and configuring its parameters each time the user is engaged in a communication session.

The architecture was also extended to allow roaming users to avail themselves through any public services provided by their current ubiquitous computing environment. The architecture leverages short-range wireless communications, and also service discovery techniques to expand the reach of the users to public devices and services available in their vicinity. We have built a prototype of the architecture and studied its performance.

Finally, we have identified the threat to users' privacy when exchanging their personal preferences and proposed a solution for protecting the privacy of the exchanged preferences during session initiation, while still be able to select the configuration parameters that maximize user's satisfaction.

1.3 Organization

The content of this thesis is organized in nine chapters. Some of these chapters have been published as conference or journal papers. Some of the ideas and experimental results in these chapters are the results of discussions and collaborations with a number of persons; I will list the contribution of each individual in here as well as in the corresponding chapter where the ideas and results where used.

- **Chapter 1: Introduction.**

  Chapter one, the current chapter, presents the motivation, context and contribution of this thesis.

- **Chapter 2: Multimedia Applications and Quality of Service (QoS) Management**

  The second chapter defines multimedia applications, their classifications and their requirements. Definition for Quality of Service (QoS) at the three levels, user, application and, network is also included. An extensive review of network-level Quality of Service techniques is presented at the end of the chapter.

- **Chapter 3: Voice-over-IP and Supporting Protocols**

  Chapter 3 introduces the Voice-over-IP Internet application, which has gained great interest in the last few years. The chapter discusses the benefits as well as the challenges to enable Voice-over-IP services over the Internet. The two major enabling signalling protocols for Voice-over-IP, mainly H.323 and SIP, are presented and compared. Other Voice-over-IP supporting protocols are also reviewed.

- **Chapter 4: A QoS Framework for Content Adaptation**

  The general framework for content adaptation is presented in Chapter 4. All the aspects involved in content adaptation are highlighted and an algorithm for selecting the QoS configuration parameters that maximize the user satisfaction is also presented. The co-op student, Bogdan Soloman, has helped out with the programming and data gathering of the QoS selection algorithm. Parts of this chapter were published in the one conference paper:

  - *K. El-Khatib, G. v. Bochmann, and A. El Saddik, "A QoS-Based Framework for Distributed Content Adaptation", First IEEE International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, Dallas, TX, Oct.2004.*

- **Chapter 5: Selecting the QoS Parameters for Multicast Applications Based on User Profile and Device Capability**

  Chapter 5 uses the framework presented in Chapter 4, but with fewer elements in an end-to-end rate-based mechanism for selecting the optimum QoS configuration

parameter values for multicast applications. The mechanism relies only on the knowledge of the preferences of the receivers and their bandwidth limitations to select the number of streams as well as the configuration settings for multicast applications. Some of the simulation results provided in this chapter are partially (around 20 %) contributed by Y. Zhong, a group member of the Distributed System Research Group (DSRG). The work has resulted in the following conference paper:

- *K. El-Khatib, G. v. Bochmann, and Y. Zhong, "Selecting the QoS Parameters for Multicast Applications Based on User Profile and Device Capability", IDMS2001, Lancaster, UK.*

- **Chapter 6: Survey of Personal Mobility Architectures**

  This chapter defines the requirements for personal mobility and examines a number of architectures that provide personal mobility.

- **Chapter 7: MobInTel: Mobile Internet Telecommunication architecture**

  Chapter 7 presents our communication service infrastructure, called the Mobile Internet Telecommunication (MobInTel) infrastructure, for supporting personal mobility over the Internet. MobInTel is an agent-based middleware that uses the framework presented in Chapter 4 and also the home directory agent concept of Mobile IP to achieve personal mobility. An extension to the MobInTel architecture to provide personal and service mobility in ubiquitous computing environment is also presented, implemented, and evaluated. N. Hadibi and Zhen Eric Zhang have contributed (around 25% both) to the implementation of the architecture. Work in this chapter has been published in the following papers:

  - *K. El-Khatib, Zhen E. Zhang, N. Hadibi, and G. v. Bochmann, "Personal and Service Mobility in Ubiquitous Computing Environments", Journal of Wireless communications and Mobile Computing, (Accepted) to appear, 2004.*
  - *K. El-Khatib, N. Hadibi, and G.v. Bochmann, "Support for Personal and Service Mobility in Ubiquitous Computing Environments", EuroPar 2003.*
  - *K. El-Khatib and G.v. Bochmann, "Agent Support for Context-Aware Services and Personal Mobility, MATA 2003.*

- **Chapter 8: Preserving the Privacy of User's Preferences in Multimedia Communication**

  In Chapter 8, we will identify the risks of privacy invasion of MobInTel user's preferences during the setup of interactive multimedia applications. We will introduce three schemes to solve the problem of protecting the user's privacy, with varying degree of complexity. A protocol based on distributed secure computation is also presented.

- **Chapter 9: Conclusion and Future Work**

  Chapter 9 presents the conclusion of our work, and outlines some interesting research topic that can be pursued in the future.

Chapter 2

Multimedia Applications and Quality of Service (QoS) Management

2.1 Introduction

Nobody can deny the fact that multimedia applications are becoming indispensable tools in many fields including academia, medicine, industry, and so many other fields. From entertainment to education, from shopping to personal communication, multimedia applications have been promising to change the way we live and do business. One of the most important enabling factors of multimedia applications is Quality of Service (QoS).

Generally speaking, QoS is a framework for describing the requirements of multimedia applications and their users, and the performance of the underlying transport infrastructure. QoS performance parameters for multimedia applications include the transmission delay, jitter, packet loss, and service availability at the network level, audio and video quality at the application level, and perceived user's satisfaction at the user's level. QoS management is the task of finding the right configuration and parameter values for all system components involved in the delivery of multimedia content.

This chapter talks about multimedia applications in general, their classification, and their requirements. We will then focus on one important issue with multimedia applications, which is the Quality of Service issue, where we will talk about different levels of Quality of Service: network, application and user level, and present some literature review of some work already done at each level.

2.2 Multimedia applications: Definition

Multimedia applications are defined as applications that are "capable of handling at least one type of continuous media in digital forms as well as static media" [1]. Static media is any type of media that does not have time dimension, such as text and pictures,

and their expressive content does not depend on their presentational style. Continuous media, in contrast, are time dependent, and must be presented in a certain well-defined timely fashion in order to be worthy. Audio and video media are examples of continuous media.

2.3 Multimedia applications: Constraints and requirements

In addition to the time constraint we have just mentioned, multimedia applications have other requirements including high data bit rate, service guarantees and group communication. We will discuss each of these requirement is details in the following sections.

2.3.1 Temporal constraints

As we mentioned earlier, the dynamic or continuous nature of the multimedia information compels some temporal constraints for the delivery. These temporal constraints are manifested through synchronization requirements of the multimedia content. Synchronization of multimedia information is required at different levels [2,3]: media layer, stream layer and object layer.

Media layer synchronization, also referred to as intra-stream synchronization, is mainly concerned with the synchronization between the single Logical Data Units (LDU) of the stream. A LDU is the smallest data unit that the application can identify inside the stream (such as a video frame or an audio data block). The media layer synchronization requirement is usually expressed in terms of latency and jitter of LDU's.

Stream layer synchronization, also know as inter-stream synchronization, is concerned with the relationship among different streams, such as the synchronization between the audio and video streams. Stream layer synchronization is required since media streams in one multimedia document might be delivered to a user along different network paths, and hence they might encounter different delay, which necessitates inter-stream synchronization at the receiver end.

The object layer synchronization, also known as inter-object synchronization, covers the synchronization between time-dependent objects such as audio streams and time-independent objects such as text and still images.

2.3.2 High data bit rate

The high data bit rate of multimedia applications is mostly associated with the audio and video media. The high-resolution HDTV format High Definition TV (HDTV) quality stream coded with MPEG-2 requires around 20 Mbps (compressed from 1.2 Gbps) [4], while a compressed MP3 stereo quality audio (compressed from CD audio, 44.1 kHz, 16 bits) requires 128 Kbps and high computation power to compress. This high data rate and high computation requirement imposes a high level of performance requirement on the end-systems as well as on the underlying transmission network. Multimedia storage servers are not only required to have large storage capacity, but also high rate delivery; workstations used for multimedia applications are also supposed to be computationally powerful to handle large volume of data. Underlying network is also supposed to provide enough bandwidth and transport quality to move the data between multimedia end-systems.

2.3.3 Service guarantees

Due to the high data bit rate and the time-constraint requirements of multimedia documents, multimedia applications require that the underlying infrastructure and the end-systems be able to provide a certain level of service guarantee in order to have an acceptable multimedia document. Service guarantee is usually obtained through a reservation mechanism and a monitoring mechanism to ensure that the service agreements are not violated. Section 2.5.1 gives a general review of network level technologies that promise to provide delivery guarantees. Application level QoS management systems are presented in Section 2.5.2.

2.3.4 Group communication

Multimedia applications that involve more than two communicating parties can benefit from an infrastructure that supports group communication such as multicast and

broadcast. This requirement is essential for group communication since point-to-point communication between individual pairs might be costly or not possible due to the large size of the group and the large volume of multimedia data. Group communications has a number of benefits:

- Reduction in system requirements since the data are read and processed only once.
- Reduction in network resources, especially when the data is transmitted only once, and replicated inside the network only when necessary.

2.4 Classes of multimedia applications

Generally speaking, multimedia applications can be classified into presentational multimedia applications, conversational multimedia applications, and messaging applications. Examples of presentational application are video-on-demand [5], and news-on-demand [6]. Tele-conference [7] and collaborative services [8] are examples of conversational multimedia application. The Seamless Personal Information Networking (SPIN) [9] is an example of multimedia messaging applications.

2.4.1 Presentational multimedia application

Presentational multimedia applications are considered as one-way multimedia applications, where multimedia data (audio, video, graphics, and/or text) is digitally stored or captured on one or more multimedia servers and streamed to the receivers over a broadband network. In these applications, receivers can specify the data they want and their preferences for the quality of the delivery, and the multimedia data flows from the media server(s) to the receiver's workstation. Typical applications include News-on-Demand, distance education, and video-on-demand. Another form of presentational application is referred to as distribution multimedia application, where the information distribution is initiated by the multimedia server, and not by the receiver as with presentational applications. Internet-TV is an example of this class of applications.

2.4.2 Conversational multimedia application

Conversational (or interactive) multimedia applications are applications in which two or more users are communicating with each other in real-time. All participants in a conversational application may send and receive real-time data at the same time. Applications such as Voice-over-IP (VoIP) and video-conference belong to this class. Due to their interactive nature, conversational multimedia applications usually pose higher QoS requirements on all system components than presentational multimedia applications do.

2.4.3 Messaging multimedia application

This class of applications covers non-real-time, asynchronous exchange of multimedia data. The data exchanged between users of this class of applications is continuous multimedia information; it usually happens through an electronic multimedia mailbox and it does not require real-time transmission. The mailbox might scan, filter, and even trans-code and deliver the data in a different format from which it was received. The universal mailbox application [10] falls into this class of applications.

Messaging multimedia applications usually do not have the real-time transport requirement of the presentational or conversational multimedia applications.

2.5 Quality of Service: Definition and terms

Vogel [11] defines the Quality of Service (QoS) as the following:

> "Quality of service represents the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application".

Most people agree that in order for the Internet to be the first choice next generation network, it has to provide certain QoS guarantees. Even though there is no widely available standard to measure the QoS of a certain network, there are some performance attributes that could be used to measure the QoS of a system.

Lu [1] presented a QoS model (Figure 1) that consists of three conceptual layers: user, application and system layer. At different layers, QoS parameters are expressed and measured differently. At the user layer, QoS parameters are more qualitative than quantitative. Typical values used at this layer include user's satisfaction expressed for instance as *unacceptable*, *bad*, *good*, *very good* and *excellent* quality. At the application layer, QoS parameters are expressed in terms of application parameters such as frame rate, resolution and color depth. The system layer consists of several sub-layers, including the operating system, the transport protocol, the secondary storage, and most importantly the underlying network sub-layer. At the network sub-layer, QoS parameters are expressed in term of throughput bandwidth, packet loss rate, packet delay, and jitter. In the following sections, we will discuss in details each of the layers, with a focus on the network sub-layer within the system layer.

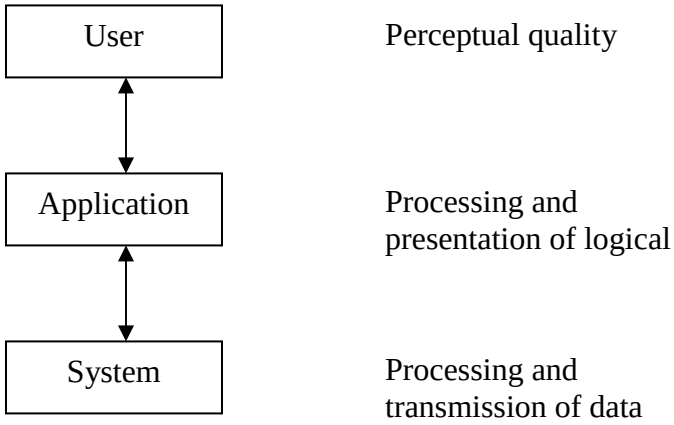

Figure 1. A conceptual model of QoS.

2.5.1 Network level QoS

When the Internet was launched almost 30 years ago, it was an experimental work to connect the U.S. Defense Department network (ARPAnet) and various other radio and satellite networks [12, 13]. The network was mainly designed to survive any partial outage, and the basic assumption was that the network itself is unreliable, and does not provide any guarantee on when it delivers the data. Today, more than 30 years later, the Internet still has the same architecture, and does not provide any guarantees on performance and reliability.

But when multimedia applications began to use the Internet as a transport medium, a number of limitations started to surface. These limitations are the result of the time constraint and shear volume of multimedia data: multimedia applications have a low tolerance for delay and jitter, and any violation of these requirements may cause a very un-acceptable degradation in the quality of these applications. Table 1 [14] shows different types of application traffic and their requirements in terms of delay, jitter and loss.

Table 1. Different types of traffic put different QoS demands on networks.

Type of Traffic	**Bandwidth**	**Delay Sensitivity**	**Jitter Sensitivity**	**Loss Sensitivity**
Bulk data transfer	10Mbps to 100Mbps	Low	None	Low
Transaction data	Less than 1 Mbps	Moderate	None	None
Voice and facsimile	8Kbps to 64Kbps	High	High	Low
Multimedia (voice plus image)	Up to 384 Kbps for video	High	Moderate	Low
Video on demand and Streaming	28.8 Kbps to 1.5 Mbps	Low	Low	Low

To meet these requirements, the Internet Engineering Task Force (IETF) has proposed a number of solutions. Notable among these solutions are the Integrated Service (IntServ), Differentiated Service (DiffServ), Resource ReSerVation Protocol (RSVP), Multi-Protocol Label Switching (MPLS), and QoS-based routing. Each solution presents a different approach to deliver network level QoS for the Internet. The following sub-sections discuss in details each of these technologies.

2.5.1.1 Integrated Service

The Integrated Service (IntServ) [15] is a standard framework designed by the IETF IntServ Working group. Its main objective is to provide, in addition to the *Best-effort* class of service, two other classes of services: the *Guaranteed* service and the *Controlled Load* (or *Predictive*) service. These classes are based on applications' requirements for delay:

- *Guaranteed* service class: provides delay-bound guarantees for applications that are critical and delay-intolerant.
- *Controlled Load* service class: provides a statistical delay bound guarantee for applications that are critical but delay-tolerant.
- *Best-effort* service class: This is the default class of service in the Internet; it has three subclasses: (1) *interactive burst* (for web similar applications); (2) *bulk* (for application like ftp); and finally (3) *asynchronous* (for applications like e-mail).

The IntServ model was built around the basic assumption that networks resources must be reserved in order to provide guarantees for application requirements, and also that the network must be signaled to make necessary reservation. Before sending data, an application must request the specific kind of service from the network. A network node required to have four components: a signaling protocol, an admission control mechanism, a packet classifier, and a packet scheduler.

An application that needs to send a data flow with some guarantees has to signal some information about the flow to the network. This information include the label of the flow (source and destination IP addresses and port numbers), the type of required service (*Guaranteed* or *Controlled Load*), the token bucket filter parameters (depth, token rate), the peak rate (p), the minimum policed unit and the maximum packet size (M). For *Guaranteed* service, the application has to also include the delay requirement. The IntServ working group has suggested the Resource ReSerVation Protocol (RSVP) (Section 2.5.1.2) as a candidate signaling protocol.

To be able to meet the required guarantees of multimedia application, the transport network must also have an admission control mechanism in place. This admission control can determine, based on the requirements of the application and the currently available transport resources, whether the network can support the requested guarantees or not. The network must also implement packet classification, policing, intelligent queuing and scheduling in order to fulfill its commitments.

The IntServ model suffers from a number of draw-backs related to its lack of scalability and the original model of the Internet [16], where each intermediate core routers of the backbone usually serves large number of flows. Such intermediate core

router would be required to have large storage capability in order to keep information about all the flows it serves. In addition, this state information also has to be kept in a high-speed access memory, since it has to be accessed for each packet. Even with current prices on memory, having large size memory would increase the price of the core router. Another problem related to this scalability issue is that InterServ model is not fault tolerant; a failure of a core router would cause all the information about the served flows to be lost, making recovery an impossible task. The second problem with the model is the large overhead incurred by implementing support mechanisms for resource reservation, admission control mechanism, packet classifier and scheduler on each intermediate core router. Running all these components on each intermediate core router contradicts with the philosophy of the Internet, where core routers are optimized to switch packet as fast as possible.

Both these drawbacks have forced the Internet community to think about other alternatives for providing different class of services; the effort has lead to the foundation of the Differentiated service model. But before we discuss the Differentiated service model, we will present the Resource ReSerVation Protocol (RSVP) which is usually considered as a companion protocol for the IntServ.

2.5.1.2 Resource ReSerVation Protocol

The Resource ReSerVation Protocol (RSVP) [16] is a signaling protocol from the Internet Engineering Task Force (IETF) that is used to reserve network resources along a path between a sender node and a receiver node. A source application can use the *path* message of RSVP to declare the specification of each flow it wishes to serve and to establish "*path state*" in the intermediate routers where multicasting is supported. Each receiver that is willing to receive the flow after receiving a *path* message can issue a *reservation* message toward the sender. *Reservation* messages travels toward the sender along the reverse path of the *path* message; intermediate routers use this message and "*path state*" to establish soft-state reservation for each flow. *Reservation* messages are not necessary forwarded to the source, since they can be terminated at the first router with sufficient resources allocated to meet the requirements of the flow reservation.

RSVP is considered as a soft-state reservation protocol, where reservations have always to be refreshed. A lifetime reservation is associated with each flow, which needs to be always refreshed as long as the flow is still going on. Once the lifetime of the flow has expired, its state is removed from the node.

Similarly to the IntServ, RSVP suffers from variable drawbacks [17], especially when it comes to the scalability issue. Intermediate routers are expected to handle a large number of flows, and hence state information and RSVP control message may impose a heavy storage requirement on these routers. This, as we mentioned earlier, contradicts with the original concept of the Internet where the state information of the flow is kept at the end-system [13], and nothing in the intermediate core routers.

2.5.1.3 Differentiated Service

To provide QoS support in the Internet at a coarser level than the IntServ, the Diffserv working group at the IETF proposed a simple and scalable architecture for service differentiation. The proposed DiffServ architecture [18] is based on the idea that applications can be classified into few categories that require different service guarantees. Data packets with different service or guarantee requirements can be tagged differently, and core router can forward each packet based on its tag.

When a packet arrives at the ingress router of a Diffserv domain, it is tagged to receive a certain forwarding treatment or per-hop behavior at every intermediate network node. This treatment is applied basically by using a queuing and scheduling mechanism that provide several classes of services. The Type of Service (TOS) octet field in the IP v.4 header or the IP v.6 Traffic Class octet can be used as a tag for the packet. Both octets are commonly referred to as Differentiated Service (DS) byte.

Diffserv is definitely more scalable than IntServ because it provides different services based on a tag carried by each packet, hence avoiding the complexity of storing per flow state information at each node. Packets are tagged only once at the entrance of the domain, reducing the overhead on intermediate core routers. Additionally, the DiffServ model does not require a signaling protocol to pass information about data flows to the network, which helps reducing the set up delay of these flows. Finally, there is no

requirement for a recovery mechanism on the core routers since they do not store any state information.

2.5.1.4 Multi-Protocol Label Switching

The Multi-Protocol Label Switching (MPLS) [19] is a forwarding scheme that evolved from Cisco's *Tag Switching*. Using the MPLS framework, a network can provide QoS by providing different classes of treatment for the packets in the network. Packets are assigned labels at the ingress of a MPLS-capable domain, and MPLS core routers can switch labels. The label of each packet determines the classification, forwarding, and services for the packet.

MPLS is usually implemented as an intermediate layer, between Layer 2 (L2, link layer) and Layer 3 (L3, network layer) in the OSI seven-layer model. Each MPLS packet has a 32-bit header, encapsulated between the link layer header and the network layer header. The header contains a 20-bit label, a 3-bit Class of Service (COS) field, a 1-bit label stack indicator and an 8-bit TTL field. An MPLS capable router has only to examine the label in the header before forwarding the packet.

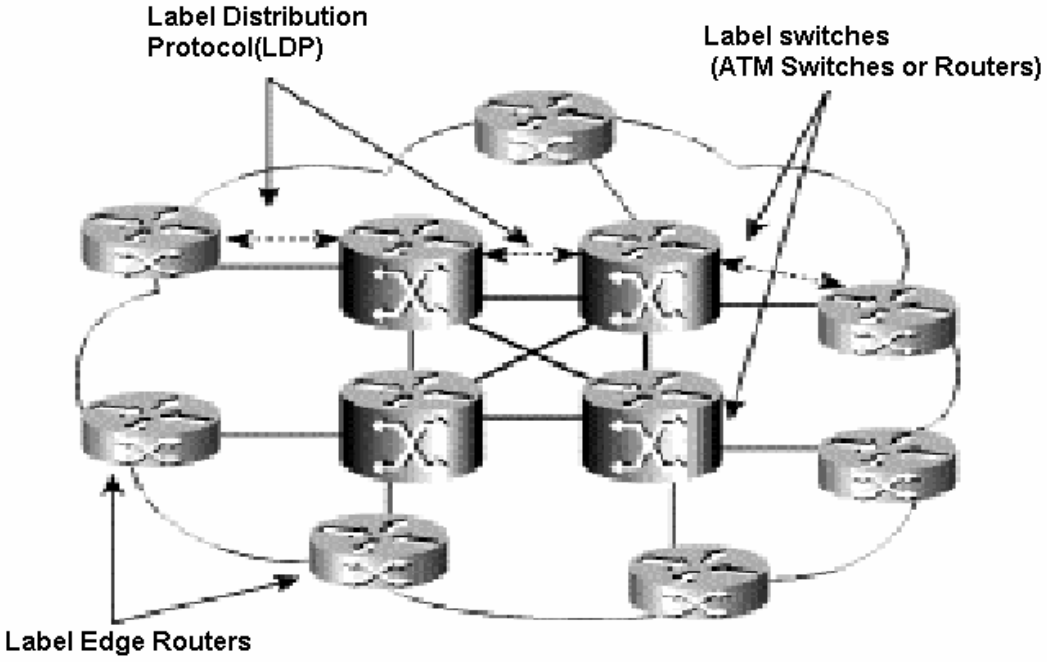

Figure 2. An MPLS enabled domain [20].

There are two types of nodes in an enabled MPLS domain: Label Edge Router (LER) and Label Switch Router (LSR). LER are located at the edge of the MPLS domain; they add or remove labels from the packets depending on the direction of the flow. LSR's are core routers that switch the labels on the packets, and forward them according to the Forward Equivalence Class (FEC) (explained later). Both LSR and LER must run a label distribution protocol in order to exchange labels for each MPLS flow. Figure 2 shows an MPLS enabled domain with both, LER's and LSR's.

In general, when a packet arrives at an MPLS enabled domain, it is assigned a certain label that determines the treatment the packet will receive on the next Label Switch Router (LSR). The label could be either embedded in the header of the data link layer (when using ATM or frame-relay at the link layer) or in the shim between the data link header and the network header. Labels have fixed size and can be easily implemented in hardware to provide higher performance. The packet is then forwarded to the next LSR, where the label is switched to another label and the packet is forwarded down the path until it exits the domain. The path followed by the packet is called a Label Switch Path (LSP) and it can be shared by all the packets that are assigned the same label. These labels are established among the LSR using the Label Distribution Protocol (LDP) [21], the Resource ReSerVation Protocol (RSVP) or even piggybacked with other routing protocol such as the Open Shortest Path First [22] or Border Gateway Protocol [23]. LDP defines four types of messages: (1) *discovery* messages, used to announce the existence of LSR in the network; (2) *session* messages, used for session management between adjacent LSR's; (3) *advertisement* messages, used for the creation and modification of FEC-to-label mapping, and finally (4) *notification* messages, used for advisory and signal error information.

The Forward Equivalence Class (FEC) is a representation for the group of packets that are given the same treatment on the LSR. This FEC could be based on any combination of the IP source address, IP destination address, source port number, destination port number, delay and jitter requirements, Class of Service (CoS), traffic engineering requirements or any other criteria. The binding of FEC-to-label is stored in the Label Information Base (LIB) table and is done only once at the ingress LER. In addition to the FEC-to-label mapping information, the LIB also contains the mapping

between incoming and outgoing label. When an LSR receives a packet, it replaces the label in the packet with the corresponding outgoing label from the LIB. Table 2 shows an example of a LIB.

Table 2. Label Information Base

Input Port	Incoming Label	Forward Equivalence Class	Output Port	Outgoing Label
1	9	128.89	1	4
2	8	171.69	0	5
…	…	…	…	…

Other than forwarding packets according to their label, MPLS can provide label merging, tunneling, and explicit or source-based routing. Label merging is used when packets arriving from different sources and destined to the same destination, have the same FEC. All these packets will be assigned the same outgoing label. Additionally, using a label stack mechanism [24], MPLS can provide tunneling by inserting a stack of labels in each packet. An LER would then pop/push a label out/into the label stack of the packet. MPLS can also be used as an explicit routing protocol [25], where an LER can compute the LSP for the packet, and insert the full path labels into the packet.

2.5.1.5 Constraint-based routing

Generally speaking, a resource reservation process consists of two phases: resource finding phase and resources reservation phases. The resource reservation phase depends on the resource finding phase to find a path with sufficient resources that meet the requirements. Finding the path is typically the task of the routing protocol.

Traditional routing protocols for the Internet (OSPF, RIP, BGP…) use a single metric such as path length, delay or cost as a selection criteria, and use the shortest path algorithm to compute the best available path between two nodes. But multimedia applications have stringent requirements, as we mentioned earlier, due to the temporal constraints and the sheer volume of data, which makes the task of finding a path that

meet requirements a very challenging task. Algorithms for finding a path satisfying a number of QoS constraints are called QoS routing algorithms.

QoS routing is a complex problem, which might not have polynomial-time routing algorithms [26]. A simple route search problem called the "shortest weighted-constrained" path was listed in [27] as an NP-complete problem. The same problem was also studied in [28], where the authors proposed two approximation algorithms to solve the problem in pseudo polynomial-time or polynomial time, but with additional constraints on the domain of length and weight values. Wang *et. al.* [26] looked at the complexity of finding a path subject to multiple constraints, and proved that the problem of finding a path subject to two or more constraints on delay, delay jitter, cost and loss probability is NP-complete. They showed though, that the combination of bandwidth and any of the four metrics is a feasible combination, and presented three polynomial-time algorithms for finding paths subject to the two constraints. Similarly, a distributed heuristic solution for the delay-constrained least-cost path problem was also proposed in [29].

Constraint-based routing has a number of Pros and Cons. On the Cons side, constraint-based routing incur additional communication and storage cost to collect, store routing information. It also incurs additional computational cost to compute paths that satisfy the given constraints. On the Pros side, constraint-based routing can improve network utilization and also deliver better quality by finding communication paths that meet the QoS requirements data flows.

2.5.2 Application level QoS

Although QoS Management architectures such as Intserv [15], Diffserv [16], and MPLS [19] have been proposed for some times, their introduction to the Internet has been slow and dreadful. These solutions have concentrated on the network and transport layers and have been judged to be cost deficient and very slow on financial return [30]. Even when new network devices are shipped with different techniques for providing QoS, only few enterprises have switched these options on. The major problem, expressed by Drucker [31], is that "you really need to be a rocket scientist to understand how these parameters work and how they interact with each other across your network".

To make QoS management simpler and comprehensible, a set of QoS aware systems [32,33,34,30,35] have been developed. These systems allow multimedia applications to adapt to the variation in QoS characteristics of the underlying network. A QoS manager of the applications considers all the different possible system configurations and selects the most appropriate configuration, depending on the desired QoS and the available resources.

For these systems, the QoS parameters reflect the data units of the media such as the frame rate, resolution, and color quality for video streams, and sampling rate, bits per sample, and loudness for audio stream quality. A QoS manager translates the application level QoS requirements into network level QoS requirements, and negotiates with the local network resource manager to reserve the required resources.

In addition to being able to adapt to variation in network resources, these systems must have a proper triggering mechanism that can determine when to initiate the adaptation process. This trigger mechanism also depends on monitoring the environment in order to detect the right conditions to trigger adaptation.

2.5.3 User level QoS

Although significant research has been carried out into network and application level QoS for multimedia communication [15,16,19], few research works have investigated the QoS and its contributing factors from the user's perspective. Traditional QoS metrics at the network level such as delay, jitter, bandwidth, buffer size and response time cannot sufficiently describe the quality of service as perceived by users.

Most researchers in the field of user level QoS agree that user level QoS parameters must not include technical details in describing QoS as perceived by the user. They also agree that there is a lot of subjectivity and context relevance associated with the user's perception of the QoS of a multimedia presentation [36]. For instance, within the RACE architecture [37], user level QoS was described in terms of: "a set of user-perceived characteristics of the performance of a service. It is expressed in user-understandable language and manifests itself as a number of parameters, all of which have either subjective or objective values". In [38], the authors defined the user-level QoS for video data as *very fast*, *fast*, *normal*, *slow* and *very slow* for the frame rate, and

very high, *high*, *medium*, low and *very low* for the resolution. A similar classification is also used in [39]. Another approach termed as "Query by example" was presented in [40], where the user determines the required QoS by viewing samples of predefined qualities.

An interesting approach for mapping between user-perceived QoS level and application QoS level and for selecting the best combination of application QoS level parameter values for communication sessions based on quantifying the user level preferences is described in [41]. The paper considers several QoS parameters (x_i) representing different aspects of the quality of various multimedia components and computes the user's appreciation of these parameters. The user's appreciation is expressed by a "satisfaction function" g_i which determines the "satisfaction" value s_i of the user as a function of the value x_i of the corresponding parameter, (i.e. $s_i = g_i(x_i)$). The range for the satisfaction function lies between zero and one (as indicated in Figure 3), which correspond to the minimum acceptable value (M) and an ideal value (I) for the parameter x_i.

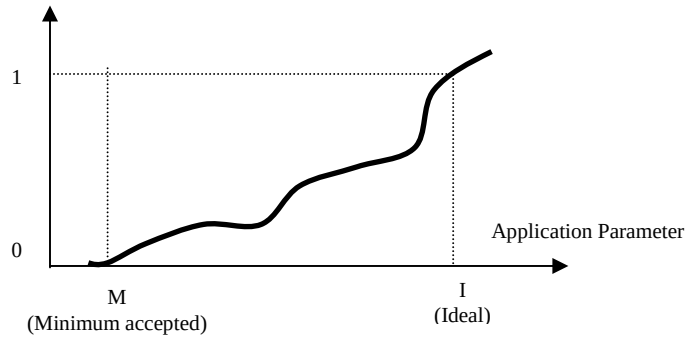

Figure 3. Example of the mapping function determining the user satisfaction depending on the value of a QoS parameter

In the case when there is more then one QoS parameter, the total satisfaction S_{tot} of the user with all the parameters was computed as the combination of the individual satisfaction functions. The authors proposed a combination function f_{comb}, which determines the total satisfaction S_{tot} from the satisfaction s_i of the individual parameters as follows:

$$S_{tot} = f_{comb}(s_1, s_2, s_3 \ldots, s_n) = \frac{n}{\sum_{i=1}^{n} \frac{1}{s_i}} \qquad \text{(Equa. 1)}$$

f_{comb} has the following properties:

- One individual low satisfaction value is enough to bring the total satisfaction to a low value. This property of the function is essentially saying that if the user is not satisfied with one of the parameters, then he/she is not satisfied in total.
- The total satisfaction of equal individual satisfactions s_i is equal to the satisfactions s_i.

In Chapter 4, we will present an extension to this approach to add weights to different satisfaction functions and to compute total satisfaction for sessions with multiple users.

In addition to the qualitative QoS metrics, most users would wish to restrict the amount of resources used in order to reduce the incurred cost. Without the notion of cost, a user would not have any incentive to specify anything less than maximum quality [42,43]. The user's specified cost is strongly related to the user's perceived quality: a user that is charged a certain amount of fee can expect to be rewarded and offered a good to excellent quality, while a user usually expect an acceptable to good quality when he/she is not charged for the service.

For the rest of the section, we will present some of the general work done on user-level QoS.

Bhatti *et. al.* [44] studied the effect of latency on the user's subjective perception of quality. The work studied the users' tolerance to delay when loading web pages in the context of e-commerce. The study found that contextual factors such as the task the user is engaged in or the method of loading the page affects the acceptance of web services. The study also found that the tolerance for delay is also influenced by the user's conceptual model of how the Internet works. For instance, users that are aware of the local caching concept were less tolerant to delay when re-visiting already visited pages. Findings from the study were incorporated in the design of web servers, in order to allow them to serve the highest number of requests and still maintain a high utility to the user.

Ghinea *et. al* [36] investigated the effect of frame rate on the user's satisfaction and on the capacity of the user to understand, analyze and synthesis the content of the presentation. The authors conducted several experiments with different frame rates and different video categories, and concluded that a reduction in frame rate does not proportionally reduce the user's understanding and perception of the content.

Mcllhagga *et. al.* [45] talked about the relation between the user's preferences and application adaptation. The authors argued that while adapting to changes in system resources is generally considered as a system level problem, the effect of the adaptation is usually visible by the user, and hence the user must always be involved in selecting the adaptation that best suit him/her. The authors presented a number of design choices that the designer of an adaptive application should consider during the design phase. These design choices include building open implementations of the application components with an interface to enable their customization. Other design issues include the degree of transparency in the adaptation and the degree of freedom in the degradation trajectories.

Hafid *et. al.* [46] stated that any QoS management function in a multimedia system should take into account the cost to be charged to the user when selecting among several possible system configurations. Users of the system were asked to specify their QoS for the requested multimedia document and the cost that they are willing to pay. In addition, users were able to specify an importance factor on each QoS parameter and on the cost also. For instance, assigning an importance value for the cost that is higher than the over-all average importance of all QoS parameters means that the user cares more about the cost charged to receive a multimedia document than about the quality of the document. Offers from the system are classified first according to whether they meet the minimum required QoS parameters into three classes: DESIRABLE, ACCEPTABLE, and CONSTRAINT. In each class, offers are then ordered according to the importance of the QoS parameters relatively to the cost of the offer.

2.6 Conclusion

The field of multimedia communication is a new field of communication and has been on the growth in the computer environments. In this chapter, we have introduced these multimedia applications, and described their classification and challenges. To

capture and provide guarantees for the requirements of multimedia applications, QoS frameworks has been used at different layers, including the system, application, and user layer. Transport QoS is a part of system QoS that has attracted wide attention in the research community. In this chapter, we have described the most promising QoS models for transport QoS, including Integrated Services, RSVP and Differentiated Services. We have highlighted their characteristics, mechanisms, and problems. We have also described MPLS and Constraint Based Routing, and do how they fit into the big QoS framework.

Chapter 3

Voice-over-IP and Supporting Protocols

Communicating voice traffic over the Internet is part of the vision to make the Internet a “one-stop network”. Many companies foresee that Voice-over-IP has the potential of reducing the cost and enabling advanced multimedia services. For the past few years, challenges have been focusing on adopting standards and on designing terminals and gateways to support Voice-over-IP. In this chapter, we will focus on the major enabling signaling protocols for Voice-over-IP, mainly H.323 and SIP. Other Voice-over-IP supporting protocols are also reviewed.

3.1 Introduction

The ability to use the Internet for carrying voice traffic, known as Voice-over-IP, has generated a lot of interest during the last decade. When it started, the main driver was to reduce or totally eliminate the cost on long distance calls over the Public Switched Telephony Network (PSTN). With Voice-over-IP (VoIP), users are able to run their voice traffic over the Internet, bypassing the long-distance carriers with their per-minute usage rates. In addition, corporations that have two separate physical networks for voice and data usually incur high cost by having support staff to maintain and support the hardware and software of two networks. Integrating both voice and data networks into one-stop packet network offers big saving opportunities for these corporations.

But the deregulation of the telecommunication industry and the advance in optical communication enabling high capacity transport have led to big cuts in the cost of long distance calls over the PSTN, and left Voice-over-IP advocates looking for new drivers to renew interest in their work. The new promises came in the easiness of the Internet as a service environment for creating and deploying new services very quickly to meet the demand of the clients. Such a service environment also has the potential to combine voice and data application to provide a new generation of services.

Research on standardizing signaling protocol for VoIP has led to the foundation of two signaling protocol: H.323 and SIP. Other supporting protocols like the Real Time Transport Protocol (RTP), Megaco and the Service Description Protocol are also described in this chapter.

3.2 Voice-over IP: What is it?

Voice-over-IP (VoIP) is an application that allows data packet networks like the Internet to transport real-time voice traffic. At the sender side, the voice signal is digitized, compressed and then divided into multiple payloads, each carried in a separate data packet. The reverse process in used at the receiver side, with some additional work to put the packets in the same order they were sent. The receiver might also need to use a buffering management technique to remove the jitter effect from the data stream.

3.3 Benefits of VoIP

The widespread of VoIP is an indication of the potential benefits of the technology. These benefits can be summarized into:

- **Cost saving**: As we mentioned earlier, one of the big motivations for VoIP was the cost saving as a result of eliminating the charges of the long-distance carrier. VoIP users may only pay a monthly flat fee for their Internet Services Providers (ISP). For big business corporations, VoIP allows them to consolidate traffic from voice and data traffic into one network, hence saving on the staffing and equipment cost.
- **Introducing new service simply and rapidly**: The Internet has changed the way service providers creates and provide services. With its openness, user-friendliness, higher flexibility and user-side customization, the Internet has made it easy and rapid to introduce new services. Providing new services on the Internet is as easy as running an application server on a machine connected to the Internet. This easiness in service provision may result in multiple service providers contending to provide similar services at competitive price resulting in cost saving and better services for the end-user.

- **Creating advanced services:** while making telephone calls and sending facsimile data over the Internet are the first applications for VoIP, it is not hard to imagine the advanced services that can provided with VoIP. The increased intelligence and highly elaborated user interface of end devices, combined with benefits of VoIP, can provide a riche environment for a large number of new services. Multimedia and multi-services applications are the first applications to benefit from VoIP. Services such as VoIP-conference, registering session content for later review, using shared whiteboard application, or even access to a call agent center from a web page, are much easier to implement with VoIP.

3.4 Issues with VoIP

While VoIP has a number of advantages such as reduced cost and bandwidth savings, it is still plagued by a number of problems that are unique to packet network, mainly the lack of QoS support in the Internet, billing, inter-working and transparency of operations.

3.4.1 Lack of QoS support in the Internet

The Internet is a packet network that was originally designed to transport data and not voice traffic. Major quality requirements for VoIP such as low delay, jitter, and packet-dropping still cannot be guaranteed over the Internet. We will next introduce each of these parameters, and outline their sources and effect on VoIP.

- **Delay**: Between the time the voice signal is sampled by the voice recorder and the time it is played by the player, voice data experiences delay from different sources including: accumulation delay, packetization delay and network delay. The accumulation delay is the time it takes the voice coder to collect enough data to make one data packet. This delay is a function of the sampling rate and the type of voice coder used. When the data for one packet is complete, the voice coder takes some time to compress the data and encapsulate it into a packet. This time is called the packetization delay. After that, the packet is sent into the network, where it incurs what is called network delay, as a result of the queuing and processing time of the

packet at each intermediate network node. Network delay is the largest delay a data packet is subject to.

A large delay value can lead to two problems: Echo and Talker Overlap. Echo happens when the round trip delay of a packet is larger than 50 milliseconds, and it results in the speaker hearing an echo of his conversation. For networks that have delay larger than the 50 milliseconds, echo cancellers are usually used. When the round trip delay is larger than 250 milliseconds, voice from both speakers will start to overlap, making the connection worse than a half-duplex connection.

- **Jitter:** Since packets in a packet-based network might travel along different paths between the sender and the receivers, successive packets might experience different delays in the network, and they might event reach the destination in a different order from the order they were sent. This variation in delay is referred to as jitter. To remove jitter, a buffer is usually used at the receiver end. Packets are stored and played with a constant delay, removing or reducing the jitter. While a buffer can reduce and even eliminate the jitter, the buffering time adds to the total delay the packet experiences, and therefore it is not a satisfying solution for interactive applications like VoIP.
- **Packet Dropping:** since the Internet uses the IP protocol which is a best effort protocol, packets might be dropped due to congestion at intermediate nodes. Transport protocols like the Transmission Control Protocol (TCP) [47] are usually used to provide transmission reliability by re-transmitting packets that are lost in the network. While data packets are not time sensitive and can be re-transmitted, voice packets cannot be retransmitted and are useful only if they arrive on time to their destination. Few techniques have been developed to mask the packet-dropping problem, including playing the last packet received for a longer time, or adding some redundant information to the stream to reconstruct lost packets. Both of these two schemes cannot solve the problem of burst packet dropping.

A number of efforts on QoS enhancements for the Internet have been and still going on. Technologies such as the ones we introduced in Chapter 2 include Integrated Service (IntServ) (Section 2.5.1.1), Differentiated Service (DiffServ) (Section 2.5.1.3), Multi-

Protocol Label Switching (MPLS) (Section 2.5.1.4), and Constraint-based routing (Section 2.5.1.5) are all promising technologies to solve the problem of QoS in the Internet. One of the major problems with these technologies is that they must be implemented in all the crossed domains between the end-points.

While these technologies provide solution in the long term, other alternative solutions such as QoS real-time reporting and QoS-based management can provide solutions for the short-term. Service providers use QoS-real time reporting to do QoS-based routing, where only the routes that provide the required QoS are selected for the service. Also QoS-real-time reporting can help to provide QoS-real time provisioning, where extra bandwidth is provided on-demand, and in real time.

Still another alternative solution to the QoS problem for VoIP is bandwidth over-provisioning. A number of service providers find that QoS-enabling technologies are not easy to understand and implement, and it is easier to add extra bandwidth to their network than building any QoS-enabling technologies into the network nodes. This direction has seen wide acceptance especially with the advance in optical networks and especially research on Dense Wave Division Multiplexing (DWDM).

3.4.2 Billing

Currently there is no standard for the tariff structure for VoIP, but something sure is that it is definitely different from that of the PSTN. In the PSTN, each connection requires the reservation of a complete circuit for the connection, and the user is charged for the duration of the connection, not based on the amount of data exchanged during the connection. While a connection based tariff structure could be used for VoIP, a per-packet tariff structure is more natural for voice over packet, since the packet is the base of communication. Implementing a per-packet billing system requires though the VoIP providers to have a more complex measuring and billing management software in place. Having such a complex billing system might not be possible because of the promise of cheap VoIP communication. Additionally, the tariff structure should also consider the case where a connection crosses different networks with different tariff structures, like between the PSTN and the Internet.

3.4.3 Inter-working and transparency of operation

The Internet is not the only type of packet network to provide VoIP service. ATM and Frame-Relay can also be used to carry voice traffic. Inter-working between all these types of networks and especially inter-working with the circuit-switched network (PSTN) raises a number of issues including data and signal trans-coding. This inter-working should also be completely transparent to the user, since what is important to the user is to get a connection with a quality comparable to the quality provided by the PSTN network, and at a lower cost.

3.5 VoIP signaling protocols

VoIP signaling protocols are needed to set up and tear down VoIP communication sessions between parties. They are also required to provide facilities to locate the users and negotiate capabilities and media encoding between the end devices. Two standards have emerged as the main signaling protocol for VoIP: H.323 [48] from the International Telecommunication Union (ITU) and Session Initiation Protocol (SIP) [49] from the Internet Engineering Task Force (IETF). We will present all the details of these protocols in the following sections, and compare them at the end.

3.5.1 H.323

H.323 is the ITU recommendation for packet based multimedia communication systems. It provides an umbrella of protocols for voice, video, and data conferencing over packet-based networks such as the Internet. Basic H.323 includes protocols for point-to-point audio communication between two terminals over a packet network that does not provide any guarantees in quality of service. H.323 includes H.225 [50] for the core messages definition, H.245 [51] for the media control, H.235 [52] as a security framework, H.450.x for supplementary services (namely call transfer and call diversion), and H.332 [53] for large group conference control.

Although basic H.323 can operate between peer H.323 terminals, other components can be used to provide additional functionalities. These components include Gatekeepers used for policy management and address resolution, Gateways that allow for network

boundary crossing, and Multipoint Control Unit (MCU) for multiparty conference control. All these components: Terminals, Gateways, Gatekeepers, and MCU are referred to as endpoints in H.323. Figure 4 shows an environment with all H.323 components. The description of each of these components is introduced below.

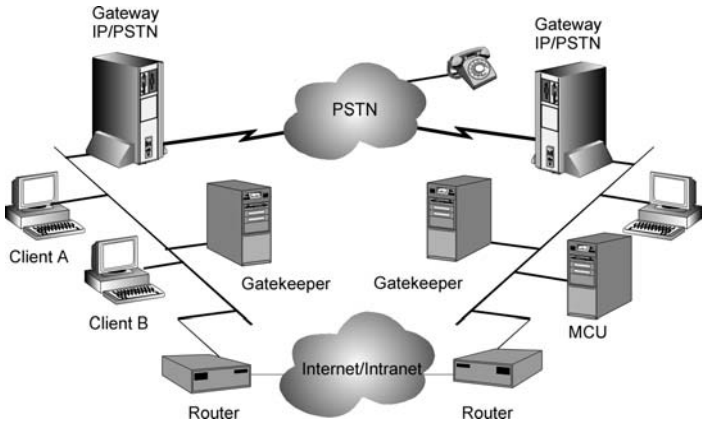

Figure 4. H.323 environment

3.5.1.1 Terminal

An H.323 terminal is any device that is capable of establishing and maintaining an H.323 connection. A terminal can be a simple black-box, an H.323 enabled telephone device, or a Personal Computer (PC). All H.323 terminals must have support for audio communication (codec G.711) with optional support for video communication. H.323 terminals must also support the H.225 call control and admission control. They must also support the H.245 protocol for capability exchange and media support.

3.5.1.2 Gatekeeper

A gatekeeper is mainly responsible for admission control and address resolution. Within each administrative domain, all H.323 endpoints can register their addresses with the gatekeeper. When a terminal is ready to place a call, it asks for the help of the gatekeeper to resolve aliases to transport addresses. If the Gatekeeper is not present, the terminal has to do this address translation itself. The gatekeeper might also enforce the domain policy during this phase. In addition, a gatekeeper may also provide additional advanced services such as dynamic routing to gateways and telephony-like services such

as *call-blocking* and *call-forwarding*; providing these services requires that all control messages exchanged between the terminals be routed through the gatekeeper.

3.5.1.3 Gateway

An H.323 gateway is an endpoint that allows H.323 devices to communicate with non-H.323 devices. Located at the crossing point between H.323 enabled domain and other non-H.323 domain such as the Public Switched Telephone Network (PSTN), a gateway provides the following functionalities:

- Call control protocols translation from H.225.0 signals to other call control signals,
- Media description translation: such as H.245 signals to other similar protocols,
- Media encoding translation, and
- Media serialization translation.

3.5.1.4 Multipoint control and processing unit

The Multipoint Control Unit (MCU) enables the establishment and control of a multiparty conference session. The MCU can be a stand-alone device, such as a workstation, or integrated into a gateway, a gatekeeper or even a terminal. It has two components: the multi-point controller (MC) and the multi-point processor (MP). The multi-point controller handles the call control and signaling for conference support. It also allows easy join-leave operations for the conference group members. The multipoint processor provides the audio and video mixing for the participants. It also provides the trans-coding functionality that allows participants with different capabilities to participate in one conference.

3.5.1.5 H.323 protocols

As we mentioned earlier, the H.323 protocol is a collection of other protocols, mainly: H.225 RAS for call signaling, registration, admission and status, H.245 for capability exchange and media control, and RTP/RTCP for data transport.

Figure 5 shows all these protocols with the underlying transport protocols. We will briefly introduce the H.225 and H.245 protocols in the following subsections. RTP/RTCP is explained later in Section 3.6.1.

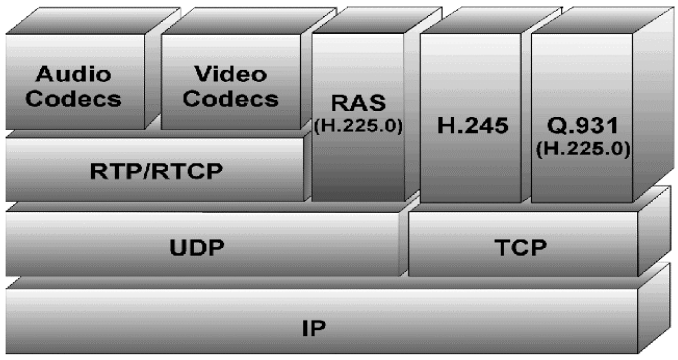

Figure 5. H.323 protocol stack

3.5.1.5.1 H.225.0: Call admission and call control

The H.225 protocol contains all the message definition required for the basic operations of H.323 endpoints. It includes two sub-protocols: Registration, Admission, and Status (RAS) protocol and the call signaling protocol derived from the Q.931[1] protocol.

3.5.1.5.2 RAS protocol: Registration, Admission, and Status

The RAS protocol is mainly used by the endpoints during the discovery of gatekeepers, endpoint registration and information exchange. An endpoint uses the RAS protocol to ask for the permission of the gatekeeper to acquire network resources. When the endpoint releases these resources, it informs the gatekeeper about the release of resources using the RAS protocol. The gatekeeper also uses the RAS protocol to acquire about the status of the endpoint.

3.5.1.5.3 Q.931: Call signaling protocol

The call signaling protocol for the H.323 protocol is derived from the Q.931 signaling protocol used in the ISDN networks with few modifications. Q.931 messages

[1] Q.931 is the control signaling protocol for the ISDN

are used between the caller and callee to exchange call setup request, call provisioning (e.g. ringing) as well as the final response to the call setup.

3.5.1.5.4 H.245: Media control protocol

Using the Q.931, H.323 terminal establish separate channel used to exchange H.245 control messages. These messages are used to negotiate and establish all the media channels between the endpoints. Messages exchanged in the H.245 protocol provide the following functionalities:

- **Master-slave determination:** in order to provide more authorities to one endpoint, such as breaking ties.
- **Capabilities exchange:** Before endpoints in H.323 agree on a common media format, they have to exchange their capability sets. This allows the endpoints to agree on number of media channels to open, the bit rate of each channel, the media encoding, and other parameters. Endpoints can re-negotiate these parameters at any time during the session.
- **Media channel control:** Before the media data is sent between the endpoints, endpoints must open one or more logical channels between each other. These logical channels are unidirectional channels, and one or more logical channel could be mapped into one transport channel.
- **Conference control:** The conference control messages allow participants of a conference to know about other participants in the conference determine a common suitable capability set for all participants and establish logical channels between all endpoints. They are also used for other functions such as floor control and chair control.

3.5.2 Session Initiation Protocol (SIP)

The Session Initiation Protocol (SIP) is the IETF signaling protocol for establishing real-time calls and conferences over packet networks. SIP is used to initiate, manage, and terminate communication sessions over the Internet. Personal mobility is a focal part of the SIP architecture. SIP also can be easily customized to provide Intelligent Network (IN) services such as call blocking, call forwarding, and call waiting. The SIP

protocol borrows a lot of techniques from other Internet protocols, such as the use of HTTP-like message format, email-like addressing scheme, and the Domain Name Service (DNS) infrastructure to locate SIP servers.

3.5.2.1 SIP main entities

The SIP architecture is a client/server architecture, where a SIP client sends a request to a SIP server, and waits for the answer. The main entities in SIP are the User Agent, the SIP Proxy Server, the SIP Redirect Server, and the SIP Registrar.

3.5.2.2 SIP User Agent

A SIP User Agent (UA) is an application that initiates and terminates SIP calls. It functions as a client, User Agent Client (UAC), when initiating SIP calls, and as a server, User Agent Server (UAS), when receiving SIP calls. UAC and UAS can communicate directly with each other, or through SIP Proxy Server and SIP Redirect Server.

3.5.2.3 SIP servers

SIP has three types of servers: SIP Proxy Server, SIP Redirect Server, and SIP Registrar. Even though a UAC can contact the UAS directly if it knows the current address of the callee, a SIP request might travel through several SIP servers before it gets to the UAS. Each intermediate server can act on the message and provide additional services.

A SIP proxy server acts in a similar fashion to an HTTP proxy server. It forwards the request from the client to another SIP server or to the User Agent. A SIP proxy server might keep some information that could be used for tracing, accounting or billing.

A SIP Redirect Server never forwards a SIP request. It always responds to a message request with a set of new addresses for the callee. The User Agent can use this information to try to communicate with the callee at any of the returned addresses.

Any SIP Proxy Server or SIP Redirect Server might use a location service to forward a request or redirect the caller to another location. Users can customize their tracking locations by continuously updating their information in the location service, independently of SIP. SIP has though one mechanism to allow SIP users to modify the

content of the location service through the SIP Registrar. A SIP Registrar is a server that accepts SIP REGISTER messages sent by SIP clients to inform the Registrar of modifications in the address. A SIP Registrar stores this information and makes it available through a location service.

3.5.2.4 SIP addressing

SIP addressing scheme is similar to the e-mail addressing scheme. A SIP address is of the form of *user@host*, where the *user* part is a user name or a telephone number. The *host* part is a domain name or a numeric IP address. Examples of SIP URL's are:

sip:information@gc.ca (host independent)
sip:information@infoserver.gc.ca (host dependent)
sip:+1-888-521-6000@nrc.gc.ca

3.5.2.5 SIP operations models

To place a call, the caller's SIP user agent client must have already the SIP URL of the callee user agent. In case the host part of the SIP URL contains an IP address, then the invitation message is sent directly to that IP address, otherwise the SIP user agent uses the service of DNS to search for the IP address of the SIP server of that domain, and then the invitation is sent to that address. A SIP server that is serving the domain where the callee resides translates a host independent address to a host dependent address and forwards the invitation to the host where the user is currently logged on. The SIP server might use the service of a location service to forward or redirect the request to another SIP server.

The basic operation model in SIP involves a SIP user agent client sending an INVITE message to a SIP user agent server. The SIP user agent server might either decline the invitation by sending an error message to the client, or accept the invitation by sending an "OK" response. In the later case, the user agent client should send an acknowledgment message "ACK" to the user agent server and the session is established. At any time, any of the two user agents can terminate the session by sending a "BYE" message.

3.5.2.5.1 Proxy operation model

As we mentioned earlier, it is possible for a SIP request to travel through one or more SIP Proxy Servers. A SIP Proxy server can use some addressing information in its cache or through the location service to forward the request to a new address. It is important to mention here that while the request and respond messages might travel several proxy servers, the data is sent directly between the caller and callee applications, without going through the same proxy servers.

3.5.2.5.2 Redirect operation model

When a SIP Redirect server receives a SIP request message, it searches its cache or contacts a location service to find information about the current location of the callee. This information is then sent back to the SIP user agent who can decide how to progress with the call.

3.5.2.6 SIP messages

SIP is a text-based protocol with a coding format very similar to the HTTP protocol. A SIP message is either a request message from the UAC to the UAS or a response message from the UAS to the UAC. Request messages are used to initiate, confirm, modify and terminate calls. They are also used to modify the registered location of the user. Table 3 shows all the request methods that could be used in a request message.

Table 3. SIP request methods

Request Method	**Usage**
INVITE	Initiate a session.
ACK	Confirm the receipt of an OK message.
BYE	Terminate a session.
CANCEL	Cancel a session during the establishment phase.
OPTIONS	Find out what are the supported features.
REGISTERED	Register a new location with the Registrar.

Response messages are sent from the UAS to the UAC to communicate either provisional responses such as “Trying” or “Ringing”, or final responses such as

"Success", "Server Busy" or "Moved Permanently". Each response message has two special header fields: the **Status-Code** field and the **Reason-Phrase** field. The **Status-Code** field contains a three-digit number that represents the result of the request message. The **Reason-Phrase** field gives the description of the status, which can be displayed to the user. Table 4 shows the entire SIP response categories with some example **Reason-Phrase.**

Table 4. Response Status-Code and Categories

Status-Code	**Category**	**Reason-Phrase example**
1xx	Informational	Trying, Ringing, Queued
2xx	Success	OK
3xx	Redirection	Moved Permanently, Moved Temporarily
4xx	Client Error	Bad Request, Method Not Allowed, Too Many Hops
5xx	Server Error	Not Implemented, Service Unavailable
6xx	Global Failure	Busy Everywhere, Does not exist anywhere

3.5.3 Comparison of H.323 and SIP

Both SIP and H.323 have gained wide acceptance in the world of VoIP. While both SIP and H.323 use RTP for the data transfer, each standard has defined its own signaling protocol stack that is completely different from the other. Below, we list some of these differences between both standards. An extensive comparison between the two protocols can be found in [54,55].

- Call setup: in contrast with SIP, H.323 v.1 and v.2 used TCP connection for the transfer of the call setup messages, and this increased the call setup delay. H.323 v.3 dropped this requirement, and hence it became similar to SIP.
- Capability set negotiation: capability set negotiation is included in both signaling protocols to ensure that both end points can understand each other's data and control signals. While the H.323 has specified the H.245 as the protocol used for exchanging capabilities between endpoints, SIP has more freedom on using any capability exchange protocol. While the Session Description Protocol (SDP) is mostly used with SIP, SIP can be the delivery protocol for any other capability exchange signaling protocol; this make SIP more generic and hence more flexible to support applications other than VoIP.

- Call setup delay: The call setup delay is defined in [54] as “the number of round trips needed for establishing audio communication between the call participants”. SIP requires 1.5 round trips to establish a session. H.323 v.1 requires 6 - 7 round trips, while H.323 v.2 reduced this number to 3 for a session that can only support G.711 audio media type. This number of round trips includes the TCP connection round trips. H.323 v.3 uses the User Datagram Protocol (UDP) [56] as the transport protocol, and so it reduced the number of round trips to 1.5 or 2.5, depending whether a gatekeeper is used or not. This is similar to the SIP call setup delay. Even though H.323 v.3 uses the UDP as the transport protocol, it establishes in parallel a TCP connection between the two endpoints, which reduces the call setup delay in case the transfer over UDP was not successful. SIP tries using UDP and TCP connections sequentially.
- Loop detection: Loop detection is used to prevent control messages from traversing the same node more than one time. SIP uses the *via* message header where each SIP server adds its address to the header. In case a server finds that its address is already included in the *via* header, the message is not forwarded, and an error response (“loop detected”) is sent back to the originating user agent. H.323 v.1 and v.2 do not have any loop detection mechanism, while H.323 v.3 has added the *PathValue* field to indicate the maximum number of gatekeeper a control message can traverse. While this mechanism does no detect loops, it can prevent infinite loops. Determining the right value for *PathValue* is still a question though.
- Protocol complexity: one of the major advantages of SIP is its simplicity that allows it to be implemented on small devices with limited computation capability. H.323 is composed of several protocols (H.225, H.245, H.235, H.246), which makes it more complex and less appealing to use on small devices.
- Conference support: Unlike the H.323, the SIP protocol does not define any conferencing control unit for conference control. A conference controller can though be easily implemented as a SIP user agent. A number of conference models supported by SIP are presented in [57].

3.6 VoIP supporting protocols

In addition to the two main signaling protocols, SIP and H.323, there are other protocols that are used to fully establish VoIP and other multimedia communication sessions. Typical examples of these protocols are the Real-time Transport Protocol (RTP) and Real-time Control Protocol (RTCP) for the data transport, the Real-time Streaming Protocol (RTSP) for controlling the delivery of streams, the Session Announcement Protocol (SAP) for advertising multimedia session, the Session Description Protocol (SDP) for the description of the multimedia sessions, and the Media Gateway Control Protocol (MGCP) to control a media gateway from a controller agent. The following sections describe each of these supporting protocols in details.

3.6.1 RTP and RTCP

The Real-time Transport Protocol (RTP) [58] is the result of the work of the Audio-Video Transport Working Group at the Internet Engineering Task Force (IETF). RTP is an Internet standard protocol that provides end-to-end data transport service that supports real-time applications. It consists of two parts: the data part and the control part. An application would use the data part to package data in RTP packets and send them to the remote application, while control information such as loss rate and inter-arrival jitter is sent using the Real-time Transport Control Protocol (RTCP) [58]; using RTCP, participants periodically exchange reports about the quality of the data delivery and the information of membership. The format of the RTP message header is shown in Figure 6.

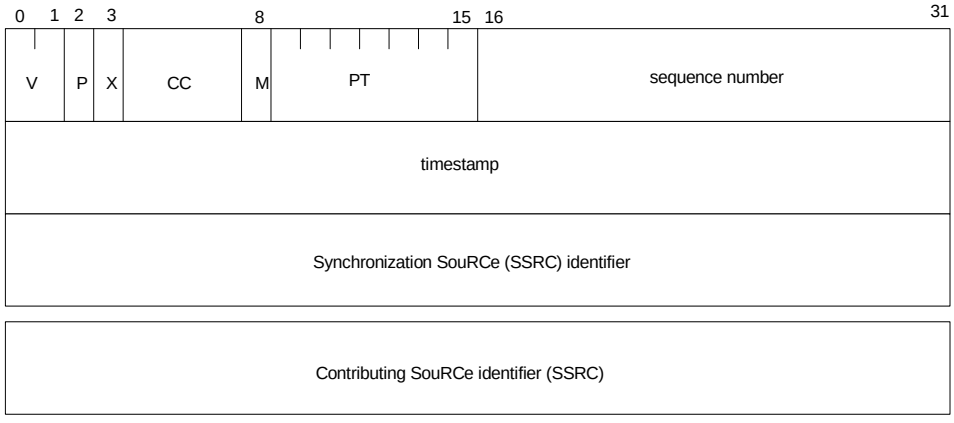

Figure 6. RTP header

The RTP runs on the top of UDP, so it does not provide any QoS guarantee or reliable delivery, but it does provide certain functionalities to support media streaming, including:

- **Time-stamping**: each RTP packet carries a timestamp that is used to synchronize different media streams belonging to the same session.
- **Sequence number**: Since RTP runs on the top of UDP, packets might be dropped or they might arrive out of order at the receiver. The sequence number inside the RTP header is used to rearrange the packets in the right order. It is also used to compute the packet loss rate that is sent back as part of the RTCP report to the sender.
- **Payload type identification**: the payload type identification is used by the sender to convey the type of the payload the packet is carrying. While the common payloads have been assigned a fixed payload type, others can be conveyed in the session control protocol.
- **Source identification**: in order to distinguish packets from different receivers, each packet carries a universally unique identifier called the Synchronization SouRCe identifier (SSRC).

3.6.2 Session Description Protocol

The Session Description Protocol (SDP) [59] is used to describe multimedia sessions for the purpose of session announcement or session invitation. SDP is based on a protocol originally used by *sd*, a conference session directory tool developed at Lawrence Berkeley Labs [60].

The SDP protocol describes basically information about the multimedia session. Information include the name and purpose of session, time(s) the session is active, the media types comprising the session, and other information describing each of these media types such as the transport address where these multimedia streams should be sent.

A multimedia session description in SDP consists of one general session description part followed by zero or more media part. The general session description part commences with the “v=” line and ends with the first media line or another “v=”

line. The media line commences with an “m=” line and ends with a “v=” or “m=” line. Some lines in the session and media description are optional while others are required. The attribute line "a=" allows SDP to be extended to include additional session description information. The order and permissible attributes in each description line is shown below; an attribute that is indicated by a “*” means that it can be used zero or more times.

Session description

v = (protocol version)

o = (owner/creator and session identifier).

s = (session name)

i =* (session information)

u =* (URI of description)

e =* (email address)

p =* (phone number)

c =* (connection information - not required if included in all media)

b =* (bandwidth information)

[One or more time descriptions (see description below)]

z =* (time zone adjustments)

k =* (encryption key)

a =* (zero or more session attribute lines)

[Zero or more media descriptions (see description below)]

Time description has the following syntax:

t = (time the session is active)

r =* (zero or more repeat times)

Media description must adhere to the following format:

m = (media name and transport address)

i =* (media title)

c =* (connection information - optional if included at session-level)

b =* (bandwidth information)

k =* (encryption key)

a =* (zero or more media attribute lines)

Even though SDP was originally designed as a complimentary protocol to the Session Announcement Protocol (Section 3.6.3) to announce multicast multimedia sessions, it is used these days to announce uni-cast session. Moreover, SDP was not designed as a negotiation protocol, since it does not have any mechanism to divide capabilities into multiple capability sets, similarly to the H.245 protocol. A successor of SDP, the Session Description and Capability Negotiation (SDPng) [61] is supposed to add the negotiation capability to the protocol.

SDP does not have its own transport protocol, but it could be used with other transport protocols such as the Session Announcement Protocol (SAP) (see Section 3.6.3), the Session Initiation Protocol (SIP) (Section 3.5.2), or the Hyper-Text Transport Protocol (HTTP).

3.6.3 Session Announcement Protocol

The Session Announcement Protocol (SAP) [62] is a protocol that users can use to periodically send sessions description, such as SDP, to a well-known multicast address and port number. Other users can listen to this multicast address to know which sessions are being announced and the descriptions of these sessions. A SAP packet consists of the SAP header, and a textual payload. The SAP header has the following fields:

- **Version**: (3 bits) indicates the version of the protocol.
- **Message Type (MT):** (3 bits) it indicates whether the SAP message is to announce a session or delete a pervious announcement.
- **E**: (1 bit) indicates whether the payload is encrypted or not.
- **C**: (1 bit) indicates whether the payload is compressed or not.
- **Authentication Length**: (8 bits) indicates the length of the authentication header.
- **Message ID hash**: (16 bits) combined with the original source field, it provides a global unique identifier for the version of the announcement.
- **Original Source**: (32 bites) contains the IP address of the source of the message.
- **Authentication header** (optional): contains the digital signature of the packet.

- **Optional payload type**: in the case when the version field is set to zero, this field is not required, and the payload of the message would be an SDP payload text; otherwise, the payload type indicates a MIME content type identifier, which specifies the format of the payload.

3.6.4 Megaco/H.248

The Megaco/H.248 [63] proposed jointly by the IETF and the ITU-T, defines an interface between a Media Gateway Controller (MGC) (also referred to as Call Agent) and the Media Gateway (MG). It is a master/slave protocol where the communication between the MGC and the MG is carried through a series of transactions. Each transaction consists of one or more commands and a mandatory response.

Megaco/H.248 allows voice, fax and multimedia calls to be switched between the public switched telephone network (PSTN) and IP, ATM or Frame Relay networks. It divides the logic of the gateway into two functional components: one for the media logic (MG) and the other for the control logic (MGC). The MGC functions include: call termination, media coding, packetization, and execution of call control stacks. For the external world, the MGC and the MG appear as a single VoIP gateway.

The main controlled entities in the MG are Termination and Context objects. A Termination is a logical entity that generates and/or terminates one or more media streams. It can accept control signals from the MGC, and it can also be programmed to detect events, which usually trigger notification messages to the MGC. The MC can also collect some statistics on a Termination object. A Context objects is an association between two or more Termination objects. Commands like *Add*, *Subtract*, and *Move*, can be used to manipulate Terminations in a Context.

3.7 Conclusion

Many experts predict that the data traffic is going to surpass traditional voice traffic. Consolidating both data and voice networks is promising to reduce communication cost and to provide myriad of new services. In this chapter, we have described the VoIP application that promises to revolutionize the field of

telecommunication. We have also presented a review of the two main enabling protocols, SIP and H.323. Additionally, we have reviewed a number of other supporting protocols.

Chapter 4

A QoS-Based Framework for Distributed Content Adaptation

4.1 Introduction

Diversity and heterogeneity of Internet clients is a major problem for multimedia delivery over the Internet. Clients range from a small single-task audio player to a complex, multi-task, multi-function desktop computer. The diversity of clients varies along different axes including display capabilities, storage space, processing power, as well as the forms of network connectivity that these clients use to access the Internet. Clients also differ in the data formats they can consume and produce, installed applications and services, and personal preferences of their users.

Today, vast amount of multimedia content already exists on the Internet. Most of this content is created and formatted for the PC users, and cannot be rendered directly on all types of client devices. Yahoo [64] and e-bay [65] have taken recently the costly approach of creating different versions of content for different access devices.

Content adaptation is considered an effective and attractive solution to the problem of mismatch in content format, device capability and user's preferences. The process of adaptation is usually applied to the sender's content in order to satisfy the device constrains of the receiver client and the preferences of its user. Possible adaptations include, but are not limited to: format change or transcoding (converting an MP3, 44 khz, 16 bits, 2 channels audio stream to a DVI, 8khz, 8 bits, 1 channel stream or reduction of image quality,), text summarization, removal of redundant information, audio to text conversion, video to key frame or video to text conversion, content extraction to list a few. While the framework presented in this chapter deals with content adaptation services in general, we will use the trans-coding type of adaptation service in the illustrations and examples.

Most currently available content adaptation schemes are best suitable for Web content. Examples of such adaptations schemes include conversion of HTML pages to

Wireless Markup Language (WML) pages, conversion of *jpeg* images to black and white *gif* images, conversion of HTML tables to plain text, or stripping of Java applets / JavaScript. These adaptation schemes do not have the same requirements and challenges of real-time multimedia content adaptations. Real-time multimedia applications involve large volumes of data making trans-coding a computationally very expensive task [66,67]. To solve this problem, some trans-coders have been implemented in hardware and can be deployed on intermediate nodes or proxies [68]. This approach cannot keep up with the pace of the constant and quick introduction of new types of clients, and requires investments in specialized hardware devices. Another more suitable approach to address the computational challenge of multimedia trans-coding is based on the observation that the general trans-coding process can be defined as combinatorial [69], and that multiple trans-coders can be chained effectively together to perform a complex trans-coding task. So, instead of having all trans-coding done by one single trans-coder, a number of trans-coders can collaborate to achieve a composite adaptation task. For instance, trans-coding a 256-color depth *jpeg* image to a 2-color depth *gif* image can be carried out in two stages: the first stage covers converting 256-color to 2-color depth, and the second step converts *jpeg* format to *gif* format. Transcoders can then be built in software and deployed easily and is shorter time to meet the needs of the users. Trans-coding would also be fast and reliable since its components can be simpler and they can also be replicated across the network.

Content adaptation plays a complimentary role to other QoS management schemes that are based on coordinating and reserving network resources [32,70,40,71,46,82]. It extends the idea of providing QoS guarantees through data manipulation and transformation.

In this chapter, we will present a general framework for content adaptation for multimedia applications. The framework identifies the major required elements for content adaptation as well as the processing model for these elements, which we call the QoS selection algorithm. Given a composite adaptation task that can be carried out in a number of stages, and given that there could be a number of possible configurations to adapt the sender's content to make it presentable at the receiver's device, the challenge is to find the appropriate chain of trans-coders that maximizes the user's satisfaction with

the final delivered content, and at the same time, derive a content that best fits the capabilities of the user's device and the communication network. The Quality of Service (QoS) selection algorithm proposed in this chapter provides personalized content by finding the most appropriate chain of trans-coders between the sender and the receiver, and also by selecting the configuration for each trans-coder. The proposed algorithm uses the user's satisfaction with the quality of the trans-coded content as the optimization metric for the path selection function. Our approach is inspired by the work of Mao *et. al* [72], in the way we construct a chain of trans-coders to match the capabilities of the sender and receiver. Our approach is different though in the way we select the sequence of trans-coders. While Mao *et. el.* used network based characteristics such as data throughput, jitter, or delay to select the trans-coders, our approach is more user centric and uses the user's satisfaction, based on the information in the user profile, as the only selection criterion. This approach is based on the observation [73,74,75,76,77] that different transport level QoS may generate similar user satisfaction, and that it is best to select a trans-coding path based on the end result, which is the user's satisfaction, and not based on single, independent, low-level factors such as delay, bandwidth, or throughput.

The rest of the chapter is organized as follows: In Section 4.2, we will advocate content adaptation as a solution for interoperability, and the different approaches used in content adaptation. Section 4.3 lists all the required elements for providing customized content adaptation. In Section 4.4, we present our methodology for using the required element from Section 4.3 to construct and optimize a directed acyclic graph of trans-coders; the algorithm for selecting the chain of trans-coders is then presented. The selection criterion for the algorithm is also introduced in Section 4.4, and finally, we end Section 4.4 with an example that shows step-by-step the results of the algorithm. Section 4.5 presents several options for discovering intermediary trans-coding services. The performance of the QoS selection algorithm is then presented in Section 4.6, and finally our conclusion is presented in Section 4.7.

4.2 Content adaptation

In today's Internet, there is a wide range of client devices in terms of both hardware and software capabilities. Device capabilities vary in different dimensions,

including processing power, storage space, display resolution and color depth, media type handling, and much more. This variety on device capabilities makes it extremely difficult for the content providers to produce a content that is acceptable and appreciated by all the client devices [78], making application-level adaptation a necessity to cover the wide population of clients. The problem is even more challenging when multicasting the content to a large number of receivers with heterogeneous device capability and preferences. In Chapter 5, we discuss this problem in more detail, and propose our solution to address the problem.

There are two main approaches for handling this diversity in content formats: static content adaptation and dynamic content adaptation, with a number of hybrids combining both approaches [79,80]. These two approaches differ in the time when the different content variants are created [81] to match the requested format. In static adaptation, the content creator generates and stores different variants of the same content on a content server, with each variant formatted for a certain device or class of devices. Hafid *et. al.* [82] presented an architecture for news-on-demand using this scheme. Static adaptation has three main advantages: (1) it is highly customized to specific classes of client devices, and (2) it does not require any runtime processing, so no delay is incurred, and (3) the content creator has the full control on how the content is formatted and delivered to the client. On the other hand, static adaptation has a number of disadvantages, mainly related to the management and maintenance of different variants of the same content [79]: (1) different content formats need to be created for each sort of device or class of devices, and needs to be re-done when new devices are introduced, and (2) it requires large storage space to keep all variants of the same content.

With dynamic content adaptation, the content is trans-coded from one format to the other only when it is requested. Depending on the location where the trans-coding takes place, dynamic content adaptation technologies can be classified into three categories: server-based, client-based, and proxy-based. In the server-based approach [83], the content server is responsible for performing the trans-coding; the content provider has all the control on how the content is trans-coded and presented to the user. Additionally, it allows the content to be trans-coded before it is encrypted, making it secure against malicious attacks. On the other hand, server-based adaptation does not

scale properly for a large number of users and requires high-end content and delivery server to handle all requests.

As for the client-based approach [84,85], the client does the trans-coding when it receives the content. The advantage of this approach is that the content can be adapted to match exactly to the characteristics of the client. But at the same time, client-based adaptation can be highly expensive in terms of bandwidth and computation power, especially for small devices with small computational power and slow network connectivity, with large volume of data might be wastefully delivered to the device to be dropped during trans-coding.

The third adaptation approach is the proxy-based approach [66,86,87,88], where an intermediary computational entity can carry out content adaptation on the fly, on behalf of the server or client. Proxy adaptation has a number of benefits including leveraging the installed infrastructure and scaling properly with the number of clients. It also provides a clear separation between content creation and content adaptation. On the other hand, some content providers may argue that they prefer to control themselves how their content is presented to the user. Also, using proxies for adaptation does not allow the use of end-to-end security solutions.

4.3 Required elements for content adaptation

Advances in computing technology have led to a wide variety of computing devices, and made interoperability very difficult. Added to this problem is the diversity of user preferences when it comes to multimedia communications. This diversity in devices and user preferences has made content personalization an important requirement in order to achieve results that satisfy the user. The flexibility of any system to provide content personalization depends mainly on the amount of information available on a number of aspects involved in the delivery of the content to the user. The more information about these aspects is made available to the system, the more the content can be delivered in a format that is highly satisfactory to the user (higher frame rate, better resolution, better audio quality,…). These relevant aspects are: user preferences, media content profile, network profile, context profile, device profile, and the profile of intermediaries (or proxies) along the path of data delivery. We will describe here each of these aspects.

User Profile: The user's profile captures the personal properties and preferences of the user, such as the preferred audio and video receiving/sending qualities (frame rate, resolution, audio quality…). Other preferences can also be related to the quality of each media types for communication with a particular person or group of persons. For instance, a customer service representative should be able to specify in his profile his/her preference to use high-resolution video and CD audio quality when talking to a client, and to use telephony quality audio and low-resolution video when communicating with a colleague at work. The user's profile may also hold the user's policies for application adaptations, such as the preference of the user to drop the audio quality of a sport-clip before degrading the video quality when resources are limited. Some other information in the user profile might also include the user's authorization, authentication and accounting information.

One of the most notable work on user profiles is the MPEG-21 standard [89], which describes attributes of the end user of multimedia content, including besides name and contact information, also content preferences, presentation preferences, accessibility and mobility preferences. These preferences are used for instance to provide effective and efficient access (search, filtering and browsing) to multimedia content. Appendix A shows the schema for the user profile we have used in the experimental prototype. This schema can be added as an extension of the MPEG-21 standard.

Content Profile: Multimedia content might enclose different media types, such as audio, video, text, and each type can have different formats [81]. Each type and format has a number of characteristics and parameters that can be used to describe the media. Such information, referred to as meta-data information, is usually included in the content profile. Some of this meta-data about the content may include:

- Information about the storage features of the content, such as the type of media (video, audio, etc), the transport protocol (RTP/UDP/IP, H.320, etc), and the format (H.261 video, MPEG video, etc).
- Information about available variants of the content, such as colored-and-black and white variants,

- Information about the author and production of the content, such as the title, and date of creation.
- Information related to the usage of the content, such as copyright, application adaptations, and usage history.

The MPEG-7 standard [90], formally named “Multimedia Content Description Interface”, offers a comprehensive set of standardized description tools to describe multimedia content. These tools allow for a complete description of what is depicted in the content, the form (coding format and size), the condition for accessing the material, the classification, the context, and the links to other relevant material. MPEG-7 also provides tools for describing variations of the content such as summaries and abstracts; scaled, compressed and low-resolution versions; and versions with different languages and modalities – audio, video, image, text, and so forth. Using the content profile, a content adaptation system can decide what type of adaptations can be applied to the content.

Context Profile: The notion of context and its implications has been a research topic for a number of research groups [91,92,93] and is still attracting more interest. According to [94] and [95], the context can be generally defined as: “any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.” Based on this definition, a context profile would include any dynamic information that is part of the context or current status of the user. Context information may include physical (e.g. location, weather, temperature), social (e.g. sitting for dinner), or organizational information (e.g. acting senior manager). Some context information, such as the role or task of the user, can be manually keyed in by the user, while other information, such as location, time of the day, weather condition, can be easily gathered using sensing devices. Some other information, such as the current status of the user, can be gathered from other sources such as the calendar of the user or from a meeting attendees list. The MPEG 21 standard includes tools for describing the natural environment characteristics of the user, including location and time, as well as the audio

and illumination characteristics of the user's environment. Resource adaptation engines can use these elements to deliver the best experience to the user.

Device Profile: To ensure that a requested content is properly rendered on the user's device, it is essential to include the capabilities and characteristics of the device into the content personalization process. Information about the rendering device may include the hardware characteristics of the device, such as the device type, processor speed, processor load, screen resolution, color depth, available memory, number of speakers, the display size, and the input and output capabilities. The software characteristics such as the operating system (vendor and version), audio and video codecs supported by the device should also be included in the device profile. The User Agent Profile (UAProf) created by the WAP Forum [96] and the MPEG 21 standard [89], both include description tools for describing device capabilities.

Network Profile: Streaming multimedia content over a network poses a number of technical challenges due to the strict QoS requirements of multimedia contents, such as low delay, low jitter, and high throughput [97]. Failing to meet these requirements may lead to a bad experience of the user [98,99]. With a large variety of wired and wireless network connectivity, it is necessary to include the network characteristics into content personalization and to dynamically adapt the multimedia content to the fluctuating network resources [100]. Achieving this requires collecting information about the available resources in the network, such as the maximum delay, error rate, and available throughput on every link over the content delivery path. A description tool for network capabilities, including utilization, delay and error characteristics are included in the MPEG 21 standard.

Profile of Intermediaries: When the content is delivered to the user across the network, it usually travels over a number of intermediaries. These intermediaries have been traditionally used to apply some added-value services, including on-the-fly content adaptations services [86,87,88]. Using intermediaries for applying adaptations alleviates the problem of clients with limited-resources [101] and overloaded servers [83].

For the purpose of content adaptation, the profile of an intermediary would usually include a description of all the adaptation services that an intermediary can provide. These services can be described using any service description language such as JINI [102], SLP [103], or WSDL [104]. A description of an adaptation service would include, for instance, the possible input and output format to the service, the required processing and computation power of the service, and maybe the cost for using the service. The intermediary profile would also include information about the available resources at the intermediary (such as CPU cycles, memory) to carry out the services. Note that the available bandwidth through an intermediary can also be included in the intermediary profile, but for clarity reasons, we have decided to include it in the network profile. Appendices B and C show the schema we have used for the trans-coding services and intermediaries.

4.4 QoS selection algorithm

When a user of a multimedia application sends a request for content to a content server in a system that supports content adaptation, the system has to decide on the required adaptations to be carried out on the content to make it satisfactory presentable to the user. This decision is affected by all the information elements introduced in Section 4.3, including the user profile, device, content, and context profile, network profile, and intermediary profile (which includes the set of available adaptation services). The QoS selection algorithm is the decision making process that analyses all these elements to find the best adaptation rules to apply to the content to make it acceptable by the user. **Figure 7** shows a high level description of the QoS selection process.

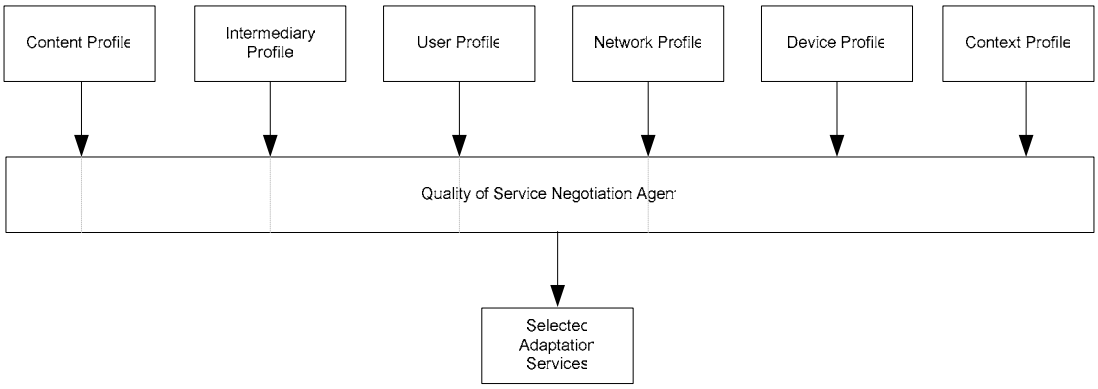

Figure 7. Framework for content adaptation

In this section, we will describe the overall QoS selection algorithm that finds the most appropriate path of trans-coders between the sender and the receiver, and also selects the configuration for each trans-coder. We will first start by defining the user's satisfaction, with constraints on the user's budget and the available network resources, as the selection criterion for the algorithm, and then show how to construct a directed graph for adaptation using the available trans-coders, the sender's content profile, and the receiver's device profile. After constructing the graph, we will show how to apply some optimization techniques on the graph to remove the extra edges in the graph, and finally present the actual QoS path and parameter selection algorithm. The QoS selection algorithm for two-party communication has been implemented as the core of the MobInTel architecture presented in Chapter 6. Using the algorithm for multimedia distribution to large groups of users is studied separately in Chapter 5.

4.4.1 User's satisfaction as selection criteria

Most Internet users are indifferent about the underlying technologies such as protocols, codecs, or resource reservation mechanisms that enable their communication session. They also are indifferent about network level QoS characteristics, such as bandwidth, delay, or throughput. All what is important for them in the end is making the communication session work in a satisfactory way: for instance, hearing without jitter and seeing without irregularity.

As we mentioned earlier, the user's preferences expressed in the user's profile can be classified as application layer QoS parameters. In order to compute the user's satisfaction with all values of the application layer configuration parameters, we have used the approach presented in [41] by Richards *et. al.*, where each application level QoS parameter is represented by a variable x_i over the set of all possible values for that QoS parameter. The satisfaction or appreciation of a user with each quality value is expressed as a satisfaction function $S_i(x_i)$. All satisfaction functions have a range of]0..1], which corresponds to the minimum acceptable (M) and ideal (I) value of x_i. The satisfaction function $S_i(x_i)$ can take any shape, with the condition that it must increase monotonically over the domain. Figure 8 shows a possible satisfaction function for the frame rate variable.

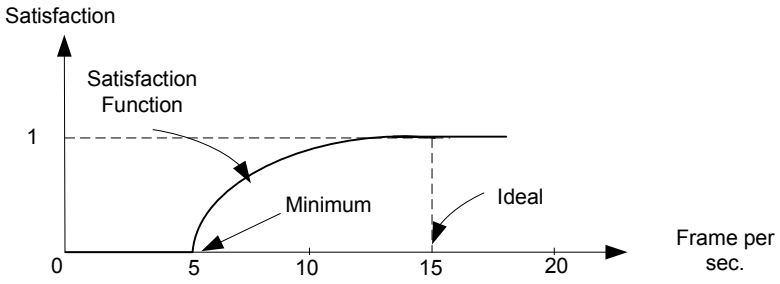

Figure 8. Possible satisfaction function for the frame rate.

In the case when there are more then one application parameter (frame rate, resolution, color depth, audio quality,…), Richards *et. al.* proposed using a combination function f_{comb} that determines the total satisfaction S_{tot} from the satisfactions s_i for the individual parameters as follows:

$$S_{tot} = f_{comb}(s_1, s_2, s_3 \ldots, s_n) = \frac{n}{\sum_{i=1}^{n} \frac{1}{s_i}} \qquad \text{(Equa. 1)}$$

The function f_{comb} has two important properties:

- *Prop. 1.* One individual low satisfaction is enough to bring the total satisfaction to a low value.
- *Prop. 2.* The total satisfaction of equal individual satisfactions s_i is equal to the satisfactions s_i.

For our prototype presented in Chapter 7, we have used the simplest case where the satisfaction function is linear between the two points of minimum acceptable and ideal values, and constant outside that interval. This means that $s(x)$ is zero for values of x smaller than M, equal to one for value of x larger than I, and linear for values in between M and I, as shown in the following equation:

$$s(x) = \begin{cases} 0 & if\ x < M \\ \dfrac{1}{I - M}(x - M) & if\ M \le x \le I \\ 1 & if\ x > I \end{cases} \qquad \text{(Equa. 2)}$$

We also note that f_{comb} is a many-to-one mapping function, and hence different combinations of individual satisfaction values are possible for one value of S_{tot}. To find out what is the best possible combination of individual satisfaction functions, another selection criterion is needed. The most reasonable selection criterion is a monetary cost threshold that specifies how much the user is willing to pay for specific media types and parameter values. Such price would include the cost for establishing the session, the cost of transfer and transformation, and in some cases, the cost of the requested data. Providing a tariff structure which determines the cost for the different values x_i of the individual application parameters, one can devise an optimization strategy for finding application parameter values that minimize the cost for a given global satisfaction S_{tot}, or maximize the satisfaction S_{tot} for a given cost value. Another selection criterion can be the available resources that can be used to provide a certain user satisfaction.

4.4.2 Extending user's satisfaction to support weighted combination and multi-user conference sessions

We think that the approach described above is a major step towards a simple user-friendly interface for user level QoS specification, however, further considerations could be taken into account as described below. A first improvement results from the observation that users in telecommunication session might find some media types more important than others. For instance, a user of a news-on-demand service might prefer to receive high quality audio with low quality video as compared to average quality audio

and average quality video. In the case of a user watching a sport event the situation may be the opposite (if the user does not care about the audio of the commenter).

This preference to individual media can play a factor when it comes to the calculation of the total satisfaction S_{tot}. By assigning different weights w_i to the different parameters x_i, S_{tot} will reflect the user preference for different media types. The combination function for the total user satisfaction can be redefined as follows:

$$S_{tot}^{user} = f_{comb}(s_1, s_2, s_3 \ldots, s_n, w_1, w_2, w_3 \ldots, w_n) = \frac{n\overline{w}}{\sum_{i=1}^{n} \frac{w_i}{s_i}} \qquad \text{(Equa. 3)}$$

where w_i is the weight for the individual satisfaction s_i and $\overline{w} = \frac{\sum_{i=1}^{n} w_i}{n}$. Equa. 3 have similar properties as Equa. 1, which is to:

- *Prop. 1.* One individual low satisfaction is enough to bring the total satisfaction to a low value.
- *Prop. 2.* The total satisfaction of equal individual satisfactions s_i with equal weight is equal to the satisfactions s_i.

These constants weight factors (AudioWeightFactor, VideoWeightFactor,..) can be selected by the user, and stored in the user profile. The selection of these weights depends on the type of service the user is willing to receive when using a specific service or communicating with a given callee.

Additionally, we have so far considered only the QoS preferences of a single user. But all conversational multimedia applications involve several users. It is therefore important to determine how the possibly conflicting preferences of the different users are reconciled in order to come up with QoS parameters that are suitable for all participating users.

In certain circumstances, some given parameters may be determined simply based on the preferences of a single user. This may be the case in a two-way teleconference between two users A and B, where the parameters of the video visible by User A would be determined based on the preferences of User A alone, and the video in the opposite

direction based on the preferences of User B. However, the situation may be more complex if the cost of the communication is paid by User A and the selection of the video received by User B has an impact on the communication cost.

In other circumstances, as for instance in the case of the joint viewing of a video clip by several participants in a teleconference, the selected quality parameters should be determined based on the preferences of all participating users. In such circumstances, we propose to use the same combination function for user satisfaction considered above and (optionally) introduce a weight for each of the participating users, called the *QoS selection weight*, which determines how much the preferences of the user influences overall QoS parameter selection. The total satisfaction (computed for all users) is then given by

$$S_{tot} = f_{comb}(s_{tot}^{usr_1}, s_{tot}^{usr_2} \ldots, s_{tot}^{usr_m}, a_1, a_2 \ldots, a_m) = \frac{m\bar{a}}{\sum_{i=1}^{m} \frac{a_i}{s_{tot}^{usr_i}}} \qquad \text{(Equa. 4)}$$

where $s_{tot}^{usr_i}$ is the total satisfaction for user *i*, and a_i is the *QoS selection weight* for user *i*. In the case that the weight of a given user is zero, the preferences of this user are not taken into account for the selection of the QoS parameters.

4.4.3 Constructing a directed graph of trans-coders

Now that we have decided on the selection criteria, the first step of the QoS selection algorithm would be to construct a directed acyclic graph (DAG) for adaptation, using the content profile, device profile, and the list of available trans-coders. Using this graph, the route selection algorithm would then determine the best path through the graph, from the sender to the receiver, which maximizes the user's satisfaction with the final received adapted content. The elements of the directed graph are the following:

1. Vertices in the graph represent intermediate trans-coders. Each vertex of the graph has a number of properties, including the computation and memory requirements of the corresponding trans-coder. Each vertex has a number of input and output links. The input links to the vertex represent the possible input formats to the trans-coder. The output links are the possible output formats of the trans-coder. Each format has a number of video parameters (video compression (H.261, H.263, MPEG-1, MPEG-2,

MPEG-4), resolution, and frame rate), and audio parameters (audio compression (G.711, G.721, DVI, MPEG-1 Layer 1, MPEG-1 Layer 2, ….), sampling rate, bits per sample, number of channels). Figure 9 shows a trans-coder T1, with two input formats, F5 and F6, and four possible output formats, F10, F11, F12 and F13. The sender node is a special case of a vertex, with only output links, while the receiver node is another special vertex with only input links.

To find the input and output links of each vertex, we rely on the information in different profiles. The output links of the sender are defined in the content profile, which includes as we mentioned earlier, meta-data information (including type and format) of all the possible variants of the content. Each output link of the sender vertex corresponds to one variant with a certain format. The input links of the receiver are exactly the input formats for the decoders available at the receiver's device. This information is available through the description of the receiver's device in the device profile. The input and output links of vertices are described in the intermediaries profile. Each intermediary profile includes the list of available trans-coders, each with the list of possible input and output formats. Each possible input format is represented as an input link into the vertex, and the output format is represented as an output link.

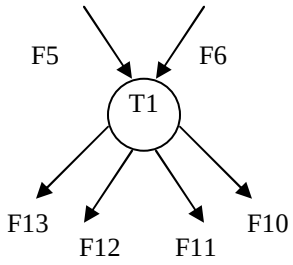

Figure 9. Trans-coder with multiple input and output links

2. Edges in the graph represent the network connecting two vertices, where the input link of one vertex matches the output link of another vertex.

To construct the adaptation graph, we start with the sender node, and then connect the outgoing edges of the sender with all the input edges of all other vertices that have the same format. The same process is repeated for all vertices. To make sure that the graph is

acyclic, the algorithm continuously verifies that all the formats along any path from the sender are distinct.

Figure 11 shows the complete pseudo-code for the graph construction algorithm, with an example of an adaptation graph shown in Figure 10. The graph is constructed with one sender, one receiver, and six intermediate vertices, each representing a trans-coder. As we can see from the graph, the sender node is connected to the trans-coder T1 along the edge labeled F5. This means that the sender S can deliver the content in format F5, and trans-coder T1 can convert this format into format F10, F11, F12, or F13. The graph optimization code fragment is explained in the next section.

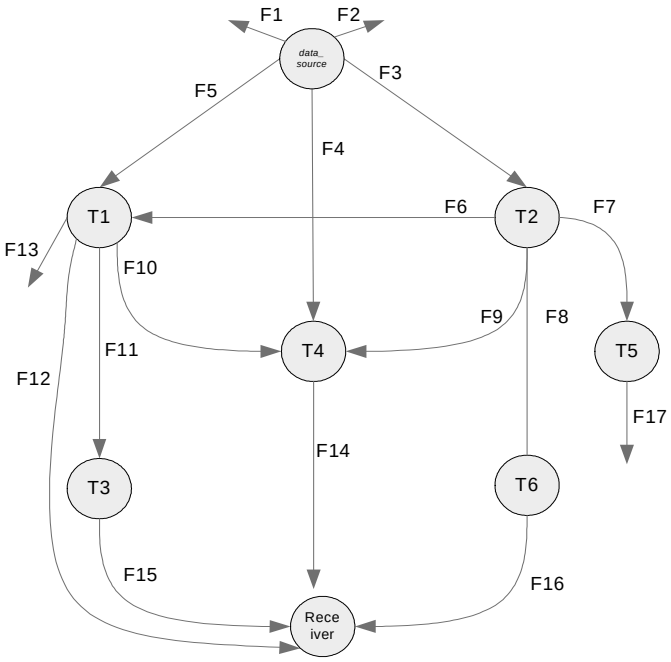

Figure 10. Directed trans-coding graph

Assuming that the resulting directed acyclic graph G(V,E) is represented with an adjacency list, the total complexity of the graph construction is $O(n^2 \lg n)$, where $n = |V|$. The time analysis of the graph construction algorithm is as follows, assuming that, on average, each node has b outgoing edges:

- Searching through all the trans-coder (lines 1-3) can be done in O(*nlog n*), since the *for* loop is repeated *n* times, and searching through the list of transcoders from the *source* node takes an average of *log n*.
- Each function call recursively calls itself n/b times

Using the Master theorem [105], we can easily compute the computational complexity of the graph construction algorithm as O(n^2 *lg n*).

```
data_source = sender's output formats;
receiver = receiver's input formats;
T = {Set of all network transcoders};
// neighbor(t) is the set of all direct children transcoders of t
// path(data_source,t) is the list of transcoders in the graph along the path from the
//   data_source to the transcoder t.

Graph_construction(){
        add_transcoder_to_graph (data_source);
}

        add_transcoder_to_graph( Transcoder t){
1           for all transc ∈ T
2               if ( (input_format(transc) = output_format(t) )     &
                  (output_format(transc) ≠ output_format(g) ∀g ∈ path(data_source,t)))
3               then neighbor(t) = neighbor(t) ∪ {transc}
4   for all ne ∈ neighbor(t)
5       add_transcoder_to_graph(ne);
        }
```

Figure 11. Pseudo-code for the graph construction algorithm

4.4.4 Graph optimization

By looking at the graph in Figure 10, we can see that there are some edges like F1, F2 or F17 that are connected only to one trans-coder. These edges cannot be a part of any path from the sender to the receiver. The same principle also applies to trans-coders other than the sender and receiver that are not on any path from the sender to the receiver. T5 is an example of a trans-coder that cannot be used to send data through it on the way from the sender to the receiver. Removing these edges and vertices help reduce the computational time of the algorithm, since it helps pruning dead-ends from the graph.

Applying optimization for the graph in Figure 10 would result in the graph shown in **Figure 13**. The pseudo-code for the graph optimization is shows in **Figure 12**.

```
        graph_optimization(Transcoder t){
6       if t ≠ receiver then {
7               for all ne ∈ neighbor(t)
8                       graph_optimization(ne);
9               if is_empty(neighbor(ne)) then
                        delete(ne);
        }
}
```

Figure 12. Pseudo-code for the graph optimization

The complexity of the graph optimization algorithm is similar to the complexity of breadth-first traversal of the graph; the graph optimization algorithm is called one time for each vertex and each edge in the graph. The total complexity of the graph optimization algorithm is $O(|V| + |E|)$.

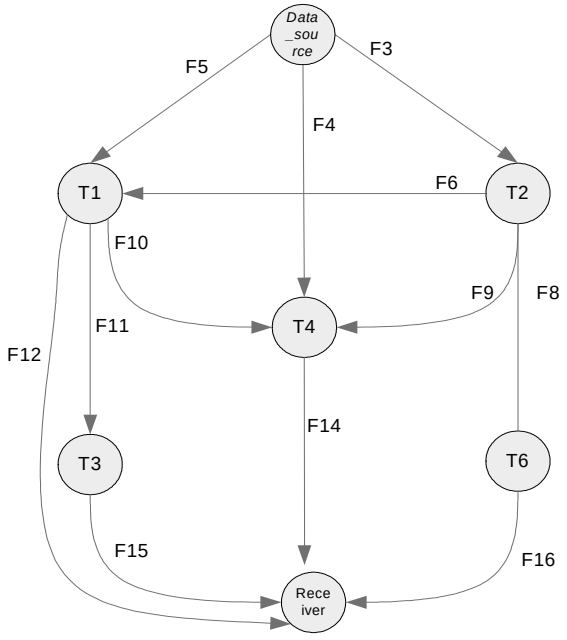

Figure 13. Optimized directed trans-coding graph

4.4.5 Adding constraints to the edges

As we have discussed earlier, the optimization criterion we have selected for the QoS selection algorithm is the user's satisfaction computed using the function f_{comb} presented in 4.4.1. The maximum satisfaction achieved by using a trans-coder T_i depends actually on a number of factors.

The first factor is the bandwidth available for the data generated by the trans-coder T_i. The more bandwidth is available to the intermediary where the trans-coder is running, the more likely the trans-coder will be able to generate trans-coded content that is more appreciate for the receiver. The available bandwidth between two trans-coders is restricted by the amount of bandwidth available between the intermediate servers where the trans-coder T_i and the next trans-coder or receiver is running. We can assume that connected trans-coders that run on the same intermediate server have an unlimited amount of bandwidth between them.

Other factors that can affect the user's satisfaction are the amount of memory and computing power required from the intermediary where the trans-coder is running, to carry out the trans-coding operation. A transcoder that requires a certain amount of memory space can not perform properly if the hosting intermediary can not allocate the minimum required memory to the transcoder. Each of these two factors is a function of the amount of input data to the trans-coder.

4.4.6 QoS selection algorithm

Once the directed acyclic adaptation graph has been constructed, the next step is to perform the QoS selection algorithm to find a chain of trans-coders, starting from the sender node and ending with the receiver node, which generates the maximum satisfaction of the receiver. Finding such as path can be similar to the problem of finding the shortest path in a directed weighted graph with similar complexity, except that the optimization criterion is the user's satisfaction, and not the available bandwidth or the number of hops.

The algorithm uses two variables representing two sets of trans-coders, the set of already considered trans-coders, called VT, and the set of candidate trans-coders, called CS, which can be added next on the partially selected path. The candidate trans-coders set

contains the trans-coders that have input edges coming from any trans-coder in the set VT. At the beginning of the algorithm, the set VT contains only the *data_source* node, and CS contains all the other trans-coders in the graph that are connected to *data_source*, and also the *receiver*. In each iteration, the algorithm selects the trans-coder T_i that , when using it, generates the highest user satisfaction. The user satisfaction is computed as an optimization function of the audio and video parameters for the output format for T_i, subject to the constraint of available bandwidth between T_i and its ancestor trans-coder, and also subject to the remaining user's budget. T_i is then added to VT. The CS set is then updated with all the neighbor trans-coders of T_i. The algorithm stops when the CS set is empty, or when the *Receiver* node is selected to be added to VT. The complete description of the algorithm is given in Figure 15. Each transcoding service or node in the created graph has the structure shown in **Figure 14**:

```
Trans-coding-service-structure{
      input_formats, output_formats: Array of Format_Type  // list of possible input and output formats
      selected_input, selected_output: Format_Type         // The selected input and output format
      transcoding_and_transmission_cost: real              // cost for using the trans-coding and
                                                           // transmission service. This cost can be
                                                           // defined by the trancoding service provider
      previous_selected_transcoder : Transcoder            // link to the previous transcoder, from the
                                                           //          source node
      accumulated_cost:   Real                             // accumulated cost for using all trancocders
                                                           // from the source to the current trasncoder
}                                                          // trans-coding_service_structure
```

Figure 14. Structure of the trans-coding service.

Step 1: // Let VT be the set of all considered trans-coders.
VT = {*data_source};*
// Let CS be the set of all direct neighbor transcoders of all transcoders in VT
CS = *neighbor*(*data_source*);
// Let *user_budget* be the amount of money the user is willing to pay

Step 2: For $\forall\ T_i \in$ CS
// Each trans-coder keeps a track of its parent trans-coder
// Let T_{prev} be the trans-coders in CS connected to T_i;
// Compute the perceived user's satisfaction for using all the trans-coders in CS, subject
// to two constraints: the remaining user budget and the available bandwidth between
// T_i and T_{prev}.
Optimize(user_profile, input_format, output_format, Sat_*T*[i],
user_budget,cost,available_bandwith)

Step 3: **if** *is_empty*(CS) **then**
// there are no more transcoders to consider and the receivers can not
// be reached from the *data_source* through any transcoding path.
TERMINATE(FAILURE)

Step 4: Select the trans-coder T_i that has the highest satisfaction value Sat_*T*[i], for the user.
CS = CS – { T_i };

Step 5: VT = VT + { T_i };

Step 6: Let T_i. previous_selected_transcoder = T_{prev};
T_i.accumulated_cost = T_i.previous.accumulated_cost + transcoding_and_transmission_cost [i];

Step 7: **if** T_i = *receiver* , **then GOTO** Step 10

Step 8: // compute the satisfaction for using all the neighboring transcoders of T_i and add them to CS
For $\forall\ T_j \in$ *neighbors*(T_i);
Optimize(user_profile, input_format, output_format, Sat_*T*[i],
user_budget,cost[i],available_bandiwth)
CS = CS $\cup$ {T_j };

Step 9: **GOTO** Step 3

Step 10: Print the reverse path from the *Receiver* to the *Sender*, by following the link *"previous"* of all transcoders, starting from the *Receiver*.

Figure 15. Pseudo-code for the route selection algorithm

As indicated in Step 2 and Step 8, the algorithm selects from CS the transcoder T_i that can generate the highest satisfaction value for the receiver. To compute the satisfaction value for each transcoder T_i in CS, the algorithm selects the QoS parameter values x_i that optimize the satisfaction function in Equa. 4, subject only to the constraint

remaining user's budget and the bandwidth availability that connects T_i to T_{prev} in VT. i.e.

$$bandwith\_requirement(x_1 .. x_n) \leq Bandwidth\_AvailableBetween(T_i , T_{prev}).$$

Since each trans-coder can only reduce the quality of the content, when the algorithm terminates, the algorithm would have computed the best path of trans-coders from the *data_source* to the *receiver,* and the user's satisfaction value computed on the last edge to the receiver node is the maximum value the user can achieve. To show this, assume that the selected path is the path $\{T_{11},\ldots T_{1n}\}$ in Figure 16. If the path $\{T_{21},\ldots T_{2m}\}$ is a better path, then T_{2m} should have converted the content into variant that is more appreciated by the user than the variant generated by T_{1n}. Since transcoders can only reduce the quality of content, all transcoders along the path $\{T_{21},\ldots T_{2m}\}$, should have also produced a content with higher satisfaction function than the variant produce by T_{1n}, and hence all these transcoders should have been selected before T_{1n}, which contradicts with the assumption.

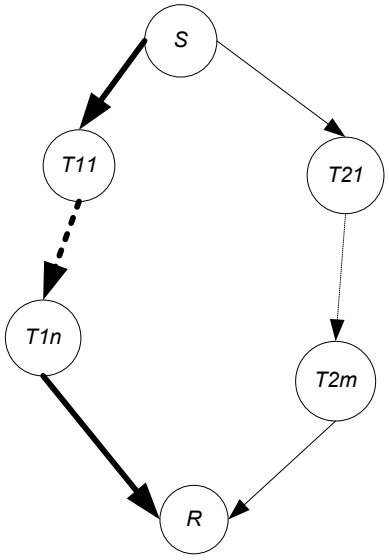

Figure 16. Graph selection

What is the running time of the QoS selection algorithm assuming an input DAG G(V,E) of trans-coding services is given? The optimization function is called one time for each edge in the graph (Step 2 or Step 8), for a total of |E| calls. Additionally, since each

vertex $v \in V$ is added at most once to the set CS, steps 3-8 are repeated at most |V| times. Each time, the transcoder with the highest user satisfaction is selected. Since we are simply storing the user's satisfaction for adding transcoder T_i in the i^{th} entry of an array, selecting the transcoder with the maximum satisfaction takes O(V) time. The total time of the algorithm is then $O(|V|^2+|E|)$.

4.4.7 Example

In this section, we will present an example to show how the route selection algorithm works. We will assume that the graph construction algorithm has generated the graph shown in Figure 17. The graph also shows the selected path with and without trans-coder T_7 as part of the graph. The selected trans-coders, user satisfaction, as well as the best current path produced by the algorithm are also shown in Table 5. Each row in the table shows the results for one iteration of the algorithm.

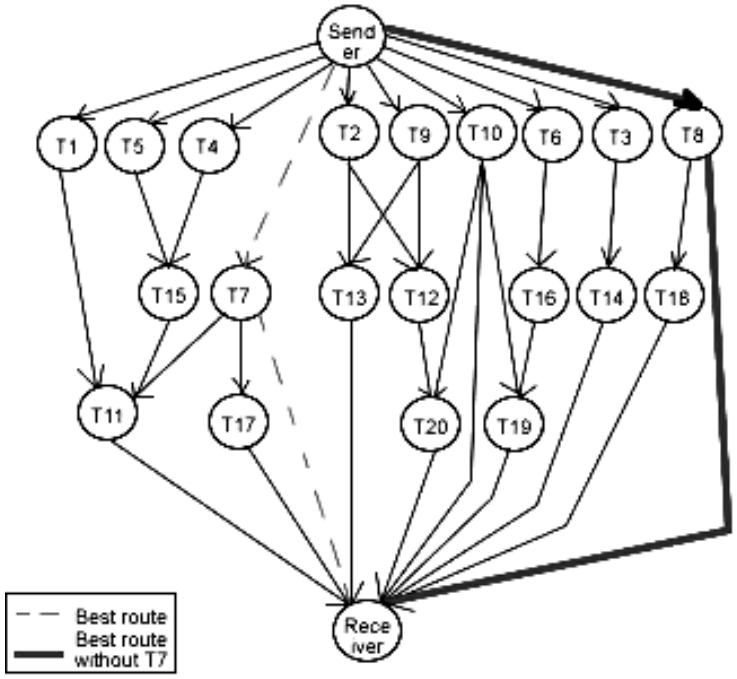

Figure 17. Example of trans-coding graph

Table 5. Results for each step of the path selection algorithm

Round	Considered Set (VT)	Candidate set (CS)	Selected trans-coder	Selected Path	Deliverd Frame Rate	User satisfaction
1	{ *data_source* }	{T1, T2, T3, T4, T5, T6, T7, T8, T9, T10}	T10	*data_source*,T10	30	1.00
2	{ *data_source*, T10}	{T1, T2, T3, T4, T5, T6, T7, T8, T9, T19, T20, *receiver*}	T20	*data_source*,T10,T20	30	1.00
3	{ *data_source*, T10, T20}	{T1, T2, T3, T4, T5, T6, T7, T8, T9, T19, *receiver*}	T5	*data_source*,T5	27	0.90
4	{ *data_source*, T10, T20, T5}	{T1, T2, T3, T4, T6, T7, T8, T9, T19, T15, *receiver*}	T4	*data_source*,T4	27	0.90
5	{ *data_source*, T10, T20, T5, T4}	{T1, T2, T3, T6, T7, T8, T9, T19, T15, *receiver*}	T3	*data_source*,T3	23	0.76
6	{ *data_source*, T10, T20, T5, T4, T3}	{T1, T2, T6, T7, T8, T9, T19, T15, T14, *receiver*}	T2	*data_source*,T2	23	0.76
7	{ *data_source*, T10, T20, T5, T4, T3, T2}	{T1, T6, T7, T8, T9, T19, T15, T14, T12, T13, *receiver*}	T1	*data_source*,T1	23	0.76
8	{ *data_source*, T10, T20, T5, T4, T3, T2, T1}	{T6, T7, T8, T9, T19, T15, T14, T12, T13, T11, *receiver*}	T11	*data_source*,T1, T11	23	0.76
9	{ *data_source*, T10, T20, T5, T4, T3, T2, T1, T11}	{T6, T7, T8, T9, T19, T15, T14, T12, T13, *receiver*}	T13	*data_source*,T2, T13	23	0.76
10	{ *data_source*, T10, T20, T5, T4, T3, T2, T1, T11, T13}	{T6, T7, T8, T9, T19, T15, T14, T12, *receiver*}	T12	*data_source*,T2, T12	23	0.76
11	{ *data_source*, T10, T20, T5, T4, T3, T2, T1, T11, T13, T12}	{T6, T7, T8, T9, T19, T15, T14, *receiver*}	T14	*data_source*,T3,T14	23	0.76
12	{ *data_source*, T10, T20, T5, T4, T3, T2, T1, T11, T13, T12, T14}	{T6, T7, T8, T9, T19, T15, *receiver*}	T8	*data_source*, T8	20	0.66
13	{ *data_source*, T10, T20, T5, T4, T3, T2, T1, T11, T13, T12, T14, T8}	{T6, T7, T9, T19, T15, *receiver*}	T7	*data_source*, T7	20	0.66
14	{ *data_source*, T10, T20, T5, T4, T3, T2, T1, T11, T13, T12, T14, T8, T7}	{T6, T9, T19, T15, *receiver*}	T6	*data_source*, T6	20	0.66
15	{ *data_source*, T10, T20, T5, T4, T3, T2, T1, T11, T13, T12, T14, T8, T7, T6}	{T9, T19, T15, *receiver*}	*receiver*	*data_source*, T7, *receiver*	20	0.66

4.5 Discovering intermediary trans-coding service

In order for the QoS selection algorithm to construct the graph of trans-coders and select the chain of these trans-coders, starting from the sender node and ending with the receiver node, the algorithm should know about available trans-coders; in other words, each trans-coder should be able to advertise the service it offers and the QoS selection algorithm should be able to discover these advertised services. This problem is known as "service advertisement and discovery". A number of approaches and protocols have been suggested to solve this problem in the ad hoc computing environment, including JINI [106], SLP [107], and WSDL [108].

The solution we have adopted for our prototype in Chapter 7 is the simple one, and is based on the assumption that the route selection algorithm has a table of all intermediate trans-coding services and their characteristics. More sophisticated approaches can be used depending on the environment where these services reside. If all these intermediate trans-coding services are located within one single administrative domain, protocols such as SLP or JINI can be used to register and discover these services. If trans-coding services are considered as computing resources in an Internet-scale Peer-to-Peer (P2P) systems then, P2P lookup protocols, such as Chord[109], CAN[110] or Pastry[111], could be used to retrieve the location and specification of instances of these services. The problem with this later approach is that the QoS selection algorithm must know what service to look for before using these protocols. Additionally, building a graph with all available trans-coding service may take a long time.

A more controllable approach, in the Internet environment, is to look for the trans-coding services that are located at a certain location inside the Internet, such as service located in domains along the data path between the communicating parties. Such approach has been used in [112]. Using this latter approach, the QoS selection algorithm may query first a local border router to get a list of all domains (BGP AS path information in the case of BGP) along the data path between the two communicating parties. Such information is usually built using inter-domain routing protocols such as the Border Gateway Protocol [113]. The graph construction algorithm would then query local directories of services located in each domain along the data path. **Figure 18** presents a

graph showing this approach, where only the trans-coders that are along the BGP path in the gray colored domains are the ones used by the graph construction algorithm. To preserve the data flow along the BGP path, the graph construction algorithm requires that each trans-coding service be assigned a *distance* value that represents the number of domains along the BGP path between the *data_source* and the trans-coder. The pseudo-code for the optimized graph construction algorithm is shown in Figure 19.

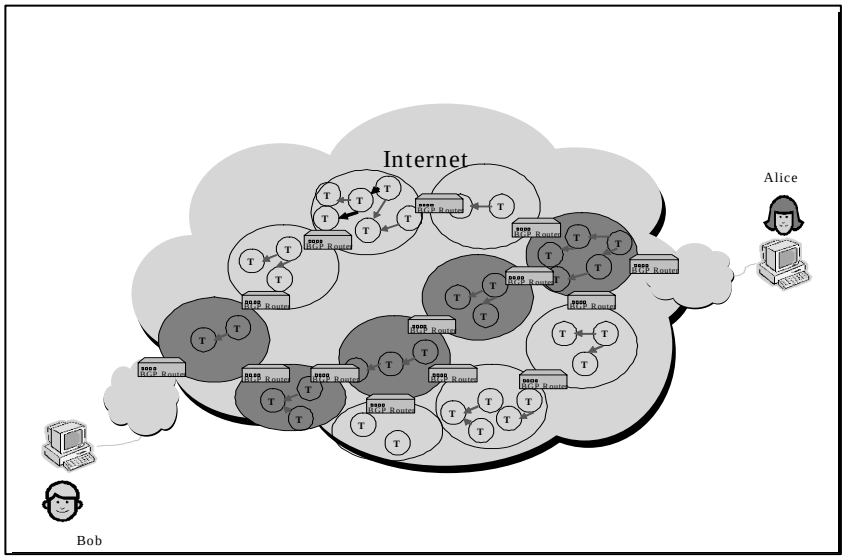

Figure 18. Trans-coders along the BGP AS path

```
        data_source = sender's output formats;
        receiver = receiver's input formats;
        T = { set of all network transcoders along the BGP path between sender's domain and the
        receiver's domain )};

        Graph_construction(){
                add_transcoder_to_graph (data_source);
        }

add_transcoder_to_graph( Transcoder t){
    for all transc ∈ T
                if ( ((distance(data_source,transc) ≥ distance(data_source, t)) &
                    (input_format(transc) = output_format(t) ) &
                    (output_format(transc) ≠ output_format(g) ∀g∈ path(data_source,t)))
            then neighbor(t) = neighbor(t) ∪ {transc}
     for all ne ∈ neighbor(t)
                add_transcoder_to_graph(ne);
}
```

Figure 19. Optimized graph construction algorithm

4.6 Performance analysis of the QoS selection algorithm

In this section, we will present some performance analysis of the QoS selection algorithm. As we mentioned, the algorithm consists of two phases: the graph construction phase and the route selection phase: the graph creation algorithm is responsible for building the DAG of trans-coders, and the route selection algorithm, is responsible for the selection of the chain of trans-coders.

We have developed the QoS selection algorithm, including the graph creation and graph optimization algorithms, using Java SDK v 1.4.2_09. We have run the code on a Pentium-4 PC, with 2.1 GHz processor, 760 MB of RAM, and running Windows XP operating system. We have simulated a network of 12 randomly connected administrative domains, and we have used between 3 and 60 transcoders randomly distributed over the 12 administrative domains.

We have run two sets of experiments: in the first set of experiments, the graph creation algorithm builds the graph from the large pool of all the trans-coders in the network. In the second set, the graph creation algorithm selected the trans-coders for the graph creation from the set of trans-coders in the administrative domains that are along the shortest BGP AS path. For both sets, we were interested in the graph creation time, route selection time, and the resulting user satisfaction. **Figure 20** and Figure 21 show the results of our experiments.

As one can see from **Figure 20**, the route selection time is negligible compared to the graph construction time. The figure shows also a big change in the graph creation time between using a large pool of trans-coders, and using just trans-coders located in administrative domains along the BGP AS path.

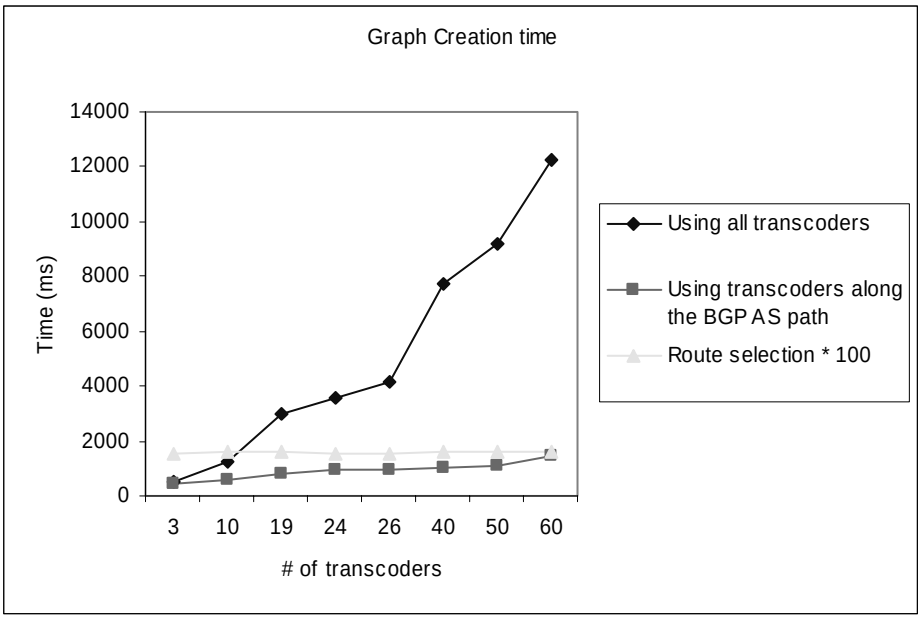

Figure 20. Graph creation time

As for the change in the user satisfaction, Figure 21 shows, as expected, a change in the value of resulting user satisfaction when using the restricted set of trans-coders along the BGP AS path. This difference is the result of reduction in the number of possible adaptation paths between the sender and the receiver.

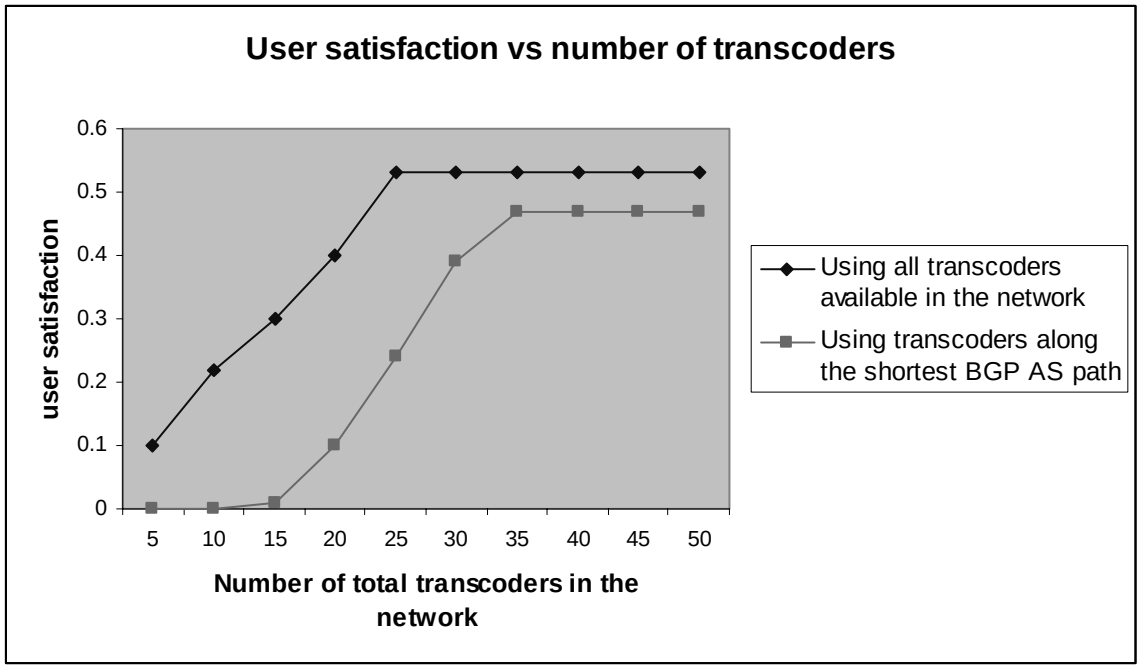

Figure 21 User satisfaction using different selection set of trans-coders.

4.7 Conclusion

Content adaptation is a natural solution to the problem of heterogeneity in client devices, network connectivity, content format, and users' preferences. This chapter presented a framework for adding several adaptation services to multimedia to make the content more satisfactory to the user. An important part of the framework is the QoS path selection algorithm that decides on the chain of adaptation services to add and the configuration parameters for each service. The theoretical and experimental analyses of the QoS path selection algorithm were also presented in this chapter.

Chapter 5

Selecting the QoS Parameters for Multicast Applications Based on User Profile and Device Capability

5.1 Introduction

Multimedia multicast applications like tele-teaching, teleconferencing, Internet TV, or remote presentation, are becoming valuable services for users of the Internet. These applications became possible due to advances in the capabilities of desktop machines and transport networks. At the level of transport, the most important advance that made these applications possible was the development of broadcast protocols, such as the Multicast Backbone (Mbone). The Mbone is an experimental virtual network imposed on the top of the Internet and has been used as a multicast test-bed. The Mbone consists of islands of multicast-capable networks, connected to each other by virtual links called "tunnels", and it shares the same transport infrastructure as the Internet.

Presentational multimedia applications are considered as one-way multimedia applications, where multimedia data is digitally stored on one or more multimedia servers and streamed to the human end user over a broadband network. In these applications, users can specify the data they want and their preferences for the quality of the delivery, and the multimedia data (audio, video, graphics, and/or text) flows from the media server(s) to the end-user workstation. Typical applications include News-on-Demand, distance education, home shopping, and video-on-demand.

Adaptive presentational applications are presentational applications that may adapt to the receiving conditions of the users. They are classified as either receiver-driven or sender-driven applications. In the receiver-driven approach, the source transmits the same data in different variants, and the receivers select the variant to receive depending on the congestion they encounter. In the sender-driven approach, the congestion feedback messages from the receivers and/or network are used to adapt the transmission rate at the source.

Based on the feedback reports, the source might adjust the sending rate [114,115,116] or keep the same sending rate and send more error-resilient data [117].For large multicast sessions, such as an Mbone broadcast or IP-based TV distribution, group sizes can be extremely large, on the order of hundreds of thousands to millions of participants. As the number of feedback messages scales with the number of receivers, the source might not be able to handle all these feedback reports, especially when these reports are sent periodically. This problem is known in the literature as feedback implosion. Several approaches have been proposed to solve the feedback implosion problem including back-off timers [118,119], probabilistic polling[120], and network aggregation [121].

In this chapter, we will show how to use the framework presented in Chapter 4 to find out what adaptations best to use at the source for generating maximum satisfaction for the receiver population. The framework uses information in the reports received from the receivers to compute the number of streams as well as the format and QoS parameters for each stream to be sent. Information in a report includes the preferences of the receiver for the QoS parameters such as frame rate and resolution and the bandwidth limit of the available local network access link. We simplify the problem of feedback implosion by assuming that the receivers will only send one report before the session starts.

Our approach differs from other research work in that the selection of the format and QoS parameters is based on the user preferences (frame rate and resolution), device capabilities, including the data format and the limitation on the bandwidth of the device rather than transport parameters, such as the data loss rate. Receivers are classified according to their bandwidth limit. Based on the capability of the source, our approach selects the number of streams as well as the format and QoS parameters of each stream. If the source has enough bandwidth to send a separate stream for each class of users, the source runs the selection algorithm for each class separately. If not, the source has to decide then on the number of streams to be sent as well as the QoS parameters for each stream. We will assume that the sender has a number of adaptation services that transcode the content into different variants, and that each stream is considered as an adapted variant of the original stream. In both cases, the selected format and QoS

parameters ensure the highest level of satisfaction of the receivers within their preferences, their device capabilities and the throughput limitation of the source. To avoid the scalability problem with the number of receivers, we introduce a way to select a virtual representative for each class, representing all receivers in that class. Receivers that have the same bandwidth limit are more likely to have similar QoS preferences and adequately represented by one virtual representative. Using group representatives, our scheme achieves content adaptation while retaining the scalability in terms of the number of receivers.

The rest of the chapter is organized as follows: Section 5.2 gives a literature review of adaptive multicast applications. Based on the throughput limit of the source, Section 5.3 details two algorithms to select to number of streams as well as the QoS parameters for each stream to send: one for the case of unlimited server throughput and the other for the case of limited server throughput In the case of unlimited server throughput, we have proved how the selection problem matches to a well known NP-hard problem. A polynomial heuristic algorithm that uses group representatives to solve the problem is also presented and evaluated in Section 5.3. Finally, we conclude in Section 5.4.

5.2 Literature review

Adaptive multicast applications are classified as either sender-driven or receiver-driven applications. In the sender-driven approach, the congestion feedback messages from the receivers and/or network are used to adapt the transmission rate at the source. In the receiver-driven approach, the source transmits the same data in different variants, and the receivers select the variant to receive depending on the congestion they encounter.

In [120], a single video stream is transmitted to all receivers, and congestion feedback is used to control the rate of the video stream. To prevent feedback implosion, a form of probabilistic feedback is used. In [122] the authors proposed an approach for the design of an available bit rate congestion control algorithm that maximizes inter-receiver fairness for multicast Available Bit Rate (ABR) sessions. Each receiver is assigned a weight value, and has an “isolated bandwidth” defined as the rate that would be achieved by the receiver when it is the only receiver in the multicast group. Every single receiver

defines his maximum acceptable loss tolerance *l* and selects its own "receiver fairness function" that maps from the actual operating bandwidth value to a fairness value. The sender receives a feedback from each receiver including the isolated bandwidth and the receiver fairness function. The sender will then try to determine the sending rate that maximizes the weighted fairness among receivers. To achieve scalability, intermediate nodes are used to aggregate feedback messages from receivers according to their isolated rates. This approach was modified to be implemented in the Internet [114]. The authors used the loss rate instead of the isolated rate to find the best sending parameters for the stream. The authors also suggested the use of two streams: a base stream with a constant data rate that can accommodate the receivers with the lowest connection rate, and another variable bit rate stream whose data rate can be modified based on feedback messages from receivers. Receivers with requests to lower the bit rate of the V-stream can always change to the base stream.

Layered encoding and group multicasting are combined in the receiver-driven approach. In the receiver-driven layered multicast (LRM) [123] approach, the video stream is decoded as a "basic" video stream, and a set of enhancement layers. Each layer is sent to a different multicast address. Receivers should receive the basic video stream and the enhancement layers that best suit their requirements. A receiver may use join-experiments to add more layers when there is extra capacity and release layers when the receiver experiences congestion. Thin Streams [124] reduces the congestion resulting from the join-experiments by dividing each layer further into "thin" layers.

The Destination Set Grouping (DSG)[115] is a hybrid between sender-driven and receiver-driven approaches. In this scheme, the source maintains a small number of video streams, broadcasting different variants of the same information. Receivers can tune to the stream with the quality they prefer. They can also send feedback messages to the source to adjust the quality of the stream to which they are tuning.

The DSG protocol is composed of two components: an *intra-stream* protocol and an *inter-stream* protocol. Using the *intra-stream* protocol, receivers can determine their status as: LOADED, UNLOADED, and CONGESTED depending on the packet loss rate. Receivers are polled in a probabilistic manner in order to estimate the number of CONGESTED and UNLOADED receivers. Depending on the fraction of CONGESTED

receivers, the source adjusts the sending rate in order to keep most of the receiver in the LOADED state.

The Source Adaptive Multi-Layered Multicast (SAMM)[116] is another hybrid algorithm that uses congestion feedback from the receivers to adjust the number of the generated layers as well as the encoding parameters of each layer. Two variations of the algorithm were proposed: a network based SAMM algorithm and an end-to-end SAMM algorithm. In the network-based algorithm, the source periodically generates a control packet called “forward feedback packet” and sends it to the multicast group. Each intermediate node updates the packet with the amount of bandwidth available for the transmission of the multicast flow. When the packet arrives at the receivers, it contains the bandwidth available on the path from the source to the receivers. Each receiver stores this value in a feedback message and sends it back to the source. Intermediate mergers combine feedback messages from downstream nodes and forward only one feedback message toward the source. If the number of requested rate values is higher than the maximum number of video layers allowed, then one or more rates must be discarded. The algorithm drops the layer with the smallest number of receivers and adds the number of receiver of that layer to the number of the preceding lower layer. The end-to-end algorithm is similar to the network-based algorithm, except that the receivers cannot determine their available bandwidth, and they only use an estimate based on the received video rate. Because the actual available bandwidth could be higher than the video arrival rate, the receiver might occasionally report a rate that is higher than the observed rate.

Active networks can also be used with adaptive multicast application [125]. In this approach, trans-coders are installed at intermediate nodes in the multicast distribution tree. The source sends only one high rate variant of the data, and intermediate nodes do the trans-coding depending on the requirements of down-stream receivers and network congestion. While this approach distributes the overhead of the source, saves bandwidth by sending only one variant of the data, and performs trans-coding only when necessary, the control and management of intermediate trans-coders is a complex problem.

5.3 Selecting QoS parameters for large groups of users

Traditional QoS parameter selection has been addressed in the context of small groups, mostly in two-party sessions or in a small group of users collaborating in a teleconference session. With such a small group, it is feasible for group members to negotiate the QoS parameters that represent best the participants' preferences. However, the negotiation algorithms for small groups do not easily scale to large groups. The task of finding a common denominator for all participants in a large group can pose interesting and challenging technical problems.

In [126], Bochmann and Yang used the DMIF session management protocol to develop a distributed QoS management framework for multicasting multimedia applications. The protocol is aimed at distributing part of the QoS management process between source and receivers; each receiver process can make certain QoS decisions based on its local context. The QoS manager in the source node determines the list of potential stream variants for each logical multimedia stream, and informs all the receivers about these variants. Based on the user profile, the QoS agent at the receiver node selects the stream that gives the highest level of appreciation to the receiver. The QoS agent can request a certain stream from the QoS manager if the stream is not currently transmitted. They used the control-plane of DMIF for the session management and illustrated its usage for the management of a tele-teaching application including different QoS alternatives for the participating users.

An approach to find a common ground for all participants in a session is to send the preferences of all participants to one node, which tries to find the QoS parameter values that generate the maximum satisfaction among all participants. In our approach, we assume that this node is the multimedia server or any node that controls the sending parameters of the source. All candidate receivers are required to send their profiles to this node. Intermediate nodes may be used to aggregate the user profiles as explained in Section 5.3.2.2. As we mentioned earlier, each profile will include, in addition to the QoS preferences of the user, the available bandwidth to the receiver. We assume that this available bandwidth is known and remains constant throughout the session.

In Chapter 4, we have presented a QoS management framework for inter-personal communication, which we have adopted from the framework presented in [41] for

selecting the QoS parameters that provide the highest satisfaction for the user. This selection algorithm assigns a value of zero to the total satisfaction of the user in the case that the satisfaction with one of the parameters is zero. We have modified the framework to assign weights to individual parameters and to include several participants in a session. The total satisfaction was computed then as the weighted satisfaction for all participants, with the same property that this satisfaction will be equal to zero if the satisfaction of any participant (with weight > 0) is equal to zero. This property ensures that all parties will participate in the session, even if the satisfaction of an individual is not at their maximum.

In the case of multicasting, we found that this framework for computing the overall satisfaction is not appropriate, since with a large population of receivers, at least one of the receivers would probably have a zero value for his satisfaction for any combination of QoS parameters, and therefore the overall satisfaction would always be zero. Instead, we decided to use a weighted sum of individual user satisfactions to determine the overall satisfaction of all receivers. Candidate receivers that have a zero satisfaction for the selected QoS parameters simply do not join the session.

5.3.1 Selecting QoS parameters with unlimited throughput in the source

Before the session starts, all receivers are required to send their reports to the source. The source then classifies the receivers into separate classes according to their bandwidth limits. If the source has throughput limit higher than the sum of all bandwidth limits of the class, it tries to find the QoS parameters for each class separately. The next section deals with the case when the server has a throughput lower than the throughput required by all classes.

For each class, the source tries to find the combination of the QoS parameters that generates the highest average satisfaction of all receivers in the class. This combination must also require lower bandwidth than the bandwidth limit of the class. The source selects the QoS parameters for every class of receivers, based on the preferences of the receivers in the class and their bandwidth limit. For instance, if the user satisfaction is defined as a function $f(.,.,.)$ of certain QoS parameters the algorithm then solves for the QoS parameters that maximize the sum of $f(.,.,.)$ over all users in the given class. The

source sends after a multicast report to all receivers, informing them of the number of streams, the QoS parameters for each stream, and the multicast address for each stream. Each receiver determines the stream that best suits his preferences and tunes to the address of that stream, as described in earlier work [126].

5.3.2 Selecting QoS parameters and channels with limited throughput in the source

It is very possible that the source receives requests to deliver a number of streams that exceed its bandwidth limitation. In this case, the source has to decide on the number of streams to send, as well as the combination of the QoS parameter for each stream. In this section, we will show how the problem of finding the best combination of stream variants for delivering media to a large group of receivers is an NP-Problem. We will start first presenting some experiments that shows how the number of channels delivered by a multimedia delivery server and the average user satisfaction changes with the change of the throughput of the server. We will then outline the problem of selecting the combination of channels to deliver, and show how the knapsack problem [127], which is known to be an NP-hard problem, maps to a special case of our problem.

We have run some experiments to see how the average satisfaction of the receivers and the number of streams change as a function of the throughput limit of the server. The results are shown in Figure 22. We selected four bandwidth limits on receivers (four classes: 28Kbps, 56Kbps, 128Kbs, unlimited), and we run the experiment with a population of 1000 receivers (Table 6). The graph shows clearly that as the bandwidth limit of the source increases, the number of streams and the average satisfaction increases until it reaches a maximum point. While the average satisfaction did not get to one (1), all users were receiving up to the maximum bandwidth of their devices. Figure 23 shows how the number of streams increased from one to four, leading to higher average satisfaction for all receivers.

Table 6. Simulated user population.

	Min. Acceptable Frame Rate	Ideal Frame Rate	Min. Acceptable Resolution	Ideal Resolution	Bandwidth Limit (Kbps)
Class 1	1..10	10..15	45x60..90x120	135x180..180x240	28
Class 2	8..17	17..22	135x180..180x240	225x300..270x360	56
Class 3	10..24	24..29	225x300..270x360	315x420..360x480	128
Class 4	12..31	31..36	315x420..360x480	405x540..450x600	Unlimited

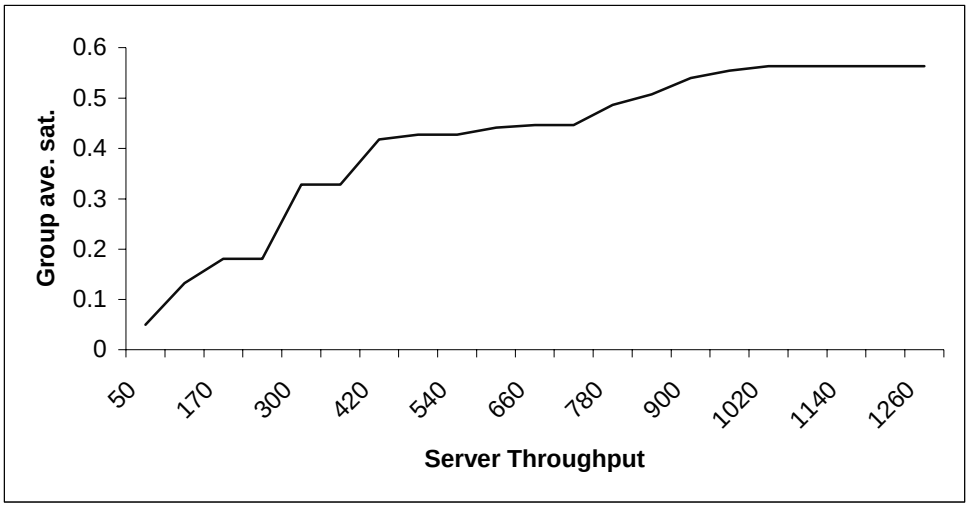

Figure 22. Server bandwidth limit vs. average satisfaction

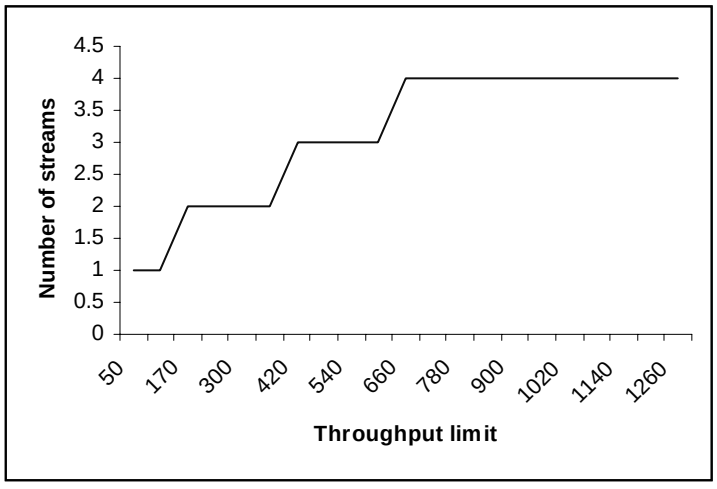

Figure 23. Server throughput limit vs. number of streams

5.3.2.1 Selecting the combination of content delivery channels: Problem definition

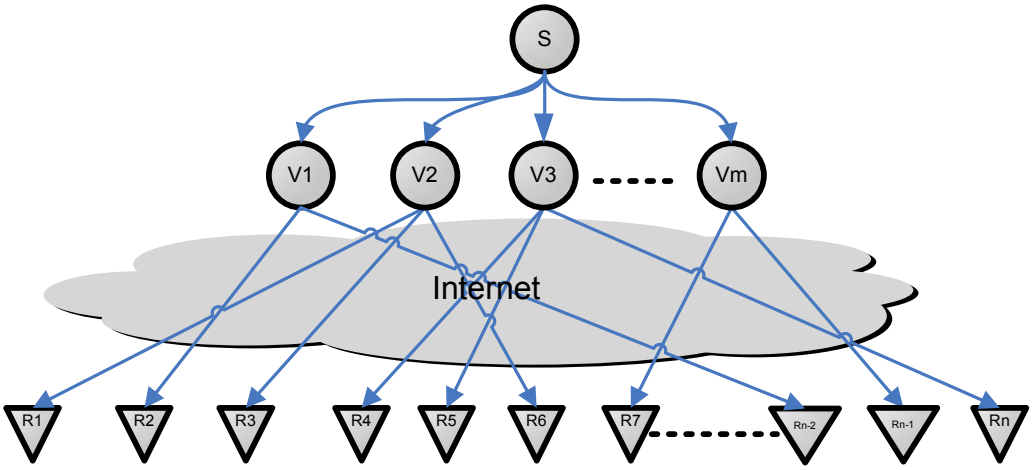

Figure 24. Assigning receivers to different variants

We will assume that the sender can, using simple or composite adaptation services, generate m variants, $V_1..V_m$. A given receiver, R_k, has the satisfaction $S_{Rk\text{-}Vi}$, when receiving variant V_i. Each variant V_c also is assigned a satisfaction, S_{Vc}, which is equal to the sum of the satisfaction of all the receivers that will receive the variant. Assume that each variant, V_i, requires a bandwidth B_{Vi} to transmit to its subscribers.We assume that the sender has a certain throughput capability, *sender_throughput*, expressed in the sender's profile . If the *sender throughput* is insufficient to send all stream variants, the problem is to select a certain number of variants that fit into the bandwidth limit and optimize the overall user satisfaction. That is, find values δ_i ($\delta_i = 1$ if V_i is selected, 0 if not) such that $\sum_{i=1}^{m} \delta_i B_{Vi} \leq$ *sender_throughput*, and $\sum_{i=1}^{m} \delta_i S_{Vi}$, is maximized.

Theorem 5.4.1 *Finding the optimal combination of variants (OCoV) that maximizes the average user satisfaction is NP-hard.*

Proof. We will prove the theorem by showing that the Knapsack problem, which is known to be NP-hard, is equivalent to a special case of the problem of finding the best combination of variants. The knapsack problem is defined as follow:

There is a knapsack of capacity c > 0 and there are N items. Each item *i* has value $val_i > 0$ and weight $w_i > 0$. Find the selection of items ($\delta_i = 1$ if item *i* is selected, 0 if not) that fit, $\sum_{i=1}^{N} \delta_i w_i \leq c$, and the total value, $\sum_{i=1}^{N} \delta_i val_i$, is maximized.

The above problem is very similar to our problem, if we consider the case where each receiver can receive data from only one variant. The sum of receiver satisfactions with each variant V_i , S_{vi}, corresponds to the value val_i, and the constant *c* corresponds to the *sender_throughput* capacity of sender. Therefore, we can see that the knapsack problem is a special case of the OCoV problem, showing that the OCoV problem is NP-hard.

5.3.2.2 Selecting the group representative

All adaptive multicast applications that require the server to solicit the receivers for feedback messages face the problem of feedback implosion, because the total number of feedback messages increases linearly with the number of receivers. Several approaches have been proposed to solve this problem including back-off timers [118, 119,], probabilistic polling [120], and selecting group representative (or feedback aggregation) [120,121,128,129].

In our application, we avoided the feedback implosion problem at two levels: first, we require that the receivers send their profiles and device limitations to the source only once, before the session starts, hence avoiding periodic feedbacks implosion. A problem with this approach is that it does not cover the variation over time of the bandwidth limit available for the receiver. To avoid this problem, users can use a conservative value for their bandwidth limit instead of the best all-time value.

To reduce further the implosion problem, we use class representation, where all receivers that have the same bandwidth limit are grouped together and are represented as one virtual receiver called the representative. The preferences of the representative are selected based on the preferences of the class members. The QoS selection algorithm would then select the QoS parameter for the class based on the QoS preferences of the representative. The decision to partition receivers according to their bandwidth limit is

based on the fact that receivers that have the same bandwidth limit are more likely to have close preferences, and the values of their preferences is more likely to represent the preferences of individual receivers.

Even though receivers in the same class are more likely to have close preferences, there is still a range on the *minimum accepted* and *ideal* preference values for the class, and this gives several possibilities for selecting the preferences of the representative. Table 7 shows different variants for selecting the *minimum accepted* and *ideal* preference of the representative receiver based respectively on the *minimum accepted* and *ideal* value for all receivers in the class.

Table 7. Variants of the preferences selection for the group representative

	***minimum accepted* value of the representative**	***ideal* value of the representative**
Variant 1	Average of the *minimum accepted* values for all receivers	Average of the *ideal* values for all receivers
Variant 2	Minimum of the *minimum accepted* values for all receivers	Minimum of the *ideal* values for all receivers
Variant 3	Minimum of the *minimum accepted* values for all receivers	Maximum of the *ideal* values for all receivers
Variant 4	Maximum of the *minimum accepted* values for all receivers	Minimum of the *ideal* values for all receivers
Variant 5	Maximum of the *minimum accepted* values for all receivers	Maximum of the *ideal* values for all receivers

To evaluate the adequacy of the class representation, we selected one class of receivers, and run the selection algorithm once with all receivers directly considered by the source for the selection of the QoS parameters of the broadcast stream (no grouping) and another time with only the preferences of the class representative considered. We compared the average satisfaction of all the receivers in these two cases. Simulation results are shown in Figure 25.

The graph in Figure 25 shows clearly that variant 2 and 4 resulted in the worse average satisfaction for the group, even though the satisfaction of the representative with its selected parameters was one (1). This is basically due to the fact that the *ideal* preference for the group representative is the minimum of the *ideal* preferences of all

receivers, reflecting hence the preferences of the most conservative receiver for the *ideal* preferences. The best variant for group representation is variant 5, where the *minimum accepted* value for the preferences of the representative is the maximum of the *minimum accepted* values for all receivers, and the *ideal* value for the representative is the maximum of all *ideal* values for all receivers. This variant avoids the conservative choice of the *minimum accepted* preference, and explores the optimism on the *ideal* preferences of all receivers.

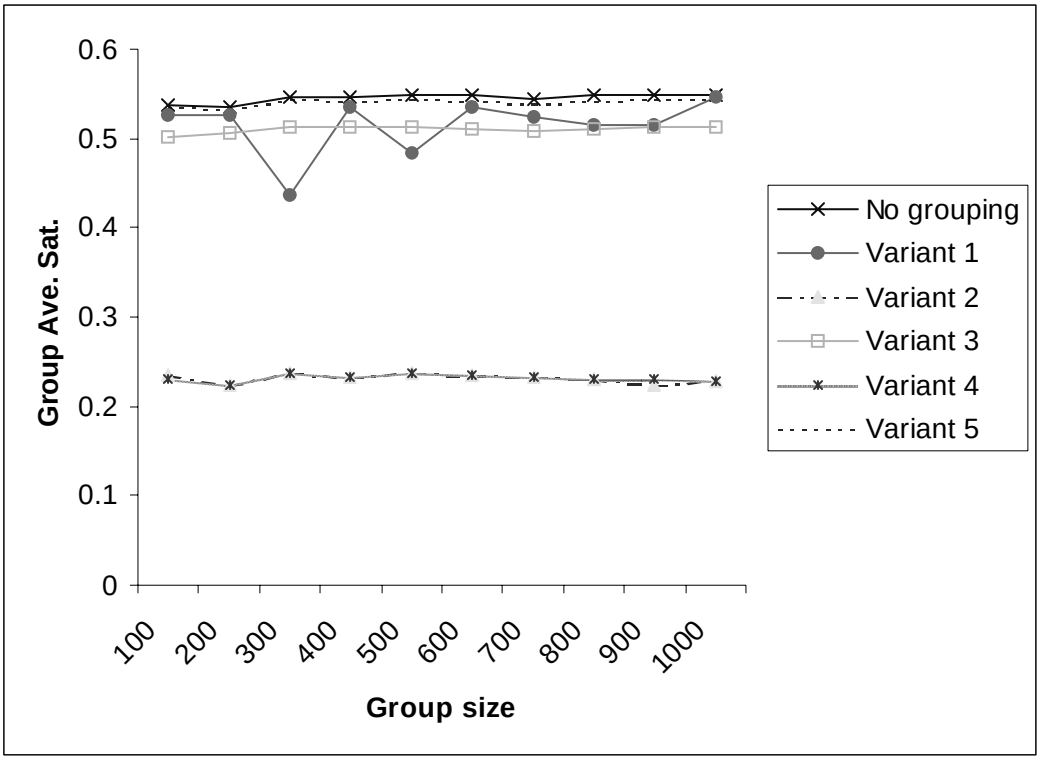

Figure 25. Average satisfaction with different variants of grouping.

5.3.2.3 A heuristic algorithm for QoS parameter and channel selection

In this section, we will present a polynomial heuristic algorithm for the *OCoV* problem. We first discuss an important practical issue neglected in the previous problem definition. Then we present a heuristic algorithm that takes this issue into account.

An important issue with selecting the number of channels and the QoS parameters for each channel with limited server throughput is that if the server decided not to transmit channel CH_i, the receivers for whom channel CH_i was the best choice, can always receive media from other channels, mainly from the channels that have close

bandwidth requirements, i.e. CH_{i-1} or CH_{i+1} (assuming that all the channels are sorted in the increasing order of their bandwidth requirements). The representative of these receivers can also help adjusting the parameters of the substitute channel in a way that guarantees better satisfaction.

We now provide a polynomial heuristic algorithm for the *OCoV* problem. The pseudo code for the algorithm is shown in Figure 26. The algorithm primarily involves the following steps: in Step 1, the server separates receivers into separate classes based on their bandwidth limit, and selects the group representative for each class in Step 2. For each class, the server then computes the QoS parameters that maximize the sum of the user satisfaction over all the users in the class. The source then tries to select the combination of channels and the QoS parameters of each channel in Step 3. In Step 5, the algorithm sorts all the channels according to their bandwidth requirements (CH_1.. CH_m). In Step 6, the algorithm repeats the process of: (a) selecting a candidate channel CH_i with the lowest total satisfaction, (b) computing the preferences for the group representative of channel CH_i, (c) checking to find the neighboring channel (CH_{i-1} or CH_{i+1}) that best accommodates the group representative of the candidate channel. (d) Finally, the candidate channel is deleted and the group representative is added to the selected neighboring class. The process is repeated until the sum of the bandwidth requirement of all the selected channels is less than or equal to the throughput of the server.

Step 1. Divide the receivers into *m* separate classes, $Class_1$ ..$Class_m$, based on their bandwidth limit
Step 2. For each *group i*, select a group representative R_i
Step 3. For each *group i*, find the QoS parameters for the channel CH_i that maximize the total satisfaction $Sat(Class_i)$ for the receivers in the group.
Step 4. $Total\_Required\_Throughput = \sum_{i=1..m} Bandwidth\_requirement(CH_i)$
Step 5. //Sort all channels according their bandwidth requirements
Sort(*Bandwith_requirement[1..m]*);
Step 6. *While (Total_Required_Throughput > Server_Throughput_Limit) do*

- Select the channel CH_i with the smallest total user satisfaction, $Sat(Class_i)$
- Compute the QoS preferences for the group representative for channel CH_i
- Check the user satisfaction of the group representative for channel CH_i with the QoS parameters of the adjacent channels, CH_{i-1} and CH_{i+1}. Select the channel *ch* which yields highest satisfaction to the group representative
- Update the total satisfaction of the channel *ch* to include the satisfaction of the group representative of group CH_i
- Remove CH_i from the list of channels
- *Total_Required_Throughput = Channel_Required_Throughput – Bandwidth(CH_i);*

End{while}

Figure 26. The pseudo code of the OCoV heuristic algorithm.

5.3.2.4 Performance evaluation of the OCoV heuristic algorithm

To study the performance of the proposed OCoV heuristic algorithm, we have run a number of simulations using the MATLAB® software. We have used the results of the simulations to compare the performance of the OCoV heuristic algorithm with the optimal algorithm. The optimal algorithm uses exhaustive search for the best combination of channels and values for the QoS value parameter that maximize the total receiver's satisfaction.

In the simulation, we have used a population of 1000 receivers with randomly generated QoS preferences, but equally divided into six classes (**Table 8**), and we have used Variant 5 from Section 5.3.2.2 to compute the group representative for each class. As we have mentioned earlier, we have used the MATLAB® software running on an IBM ThinkPad T40 laptop, with a 1500 MHz Intel® Pentium® M processor and running the Windows XP operating system.

Table 8. Population distribution

	Min. Acceptable Frame Rate	Ideal Frame Rate	Min. Acceptable Resolution	Ideal Resolution	Bandwidth Limit (Kbps)
Class 1	6..16	11..21	45x60..90x120	135x180..180x240	28
Class 2	8..18	13..23	135x180..180x240	225x300..270x360	56
Class 3	10..20	15..25	225x300..270x360	315x420..360x480	128
Class 4	12..22	17..27	315x420..360x480	405x540..450x600	256
Class 5	14..24	19..29	405x540..450x600	495x660..540x720	512
Class 6	16..26	21..31	510x680..540x720	585x780..630x840	Unlimited

The results that we were interested in were the running time of the optimal algorithm and of the OCoV heuristic algorithm, as well as the percentage of the difference between the achieved total satisfaction for the two algorithms. These simulation results for these two values are depicted in Figure 27 and Figure 28.

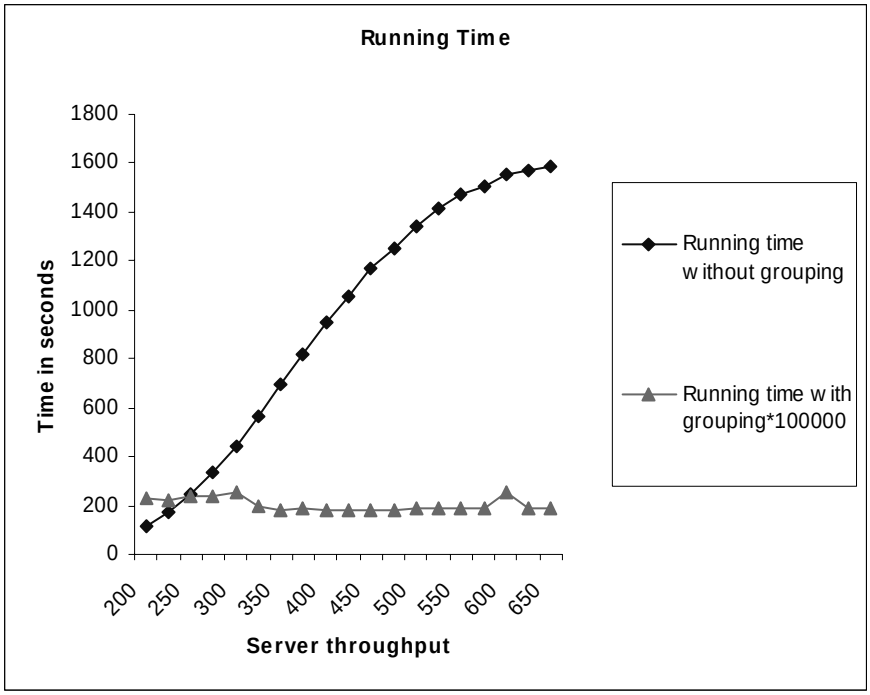

Figure 27. Running time for the OCoV heuristic algorithm and the optimal algorithm.

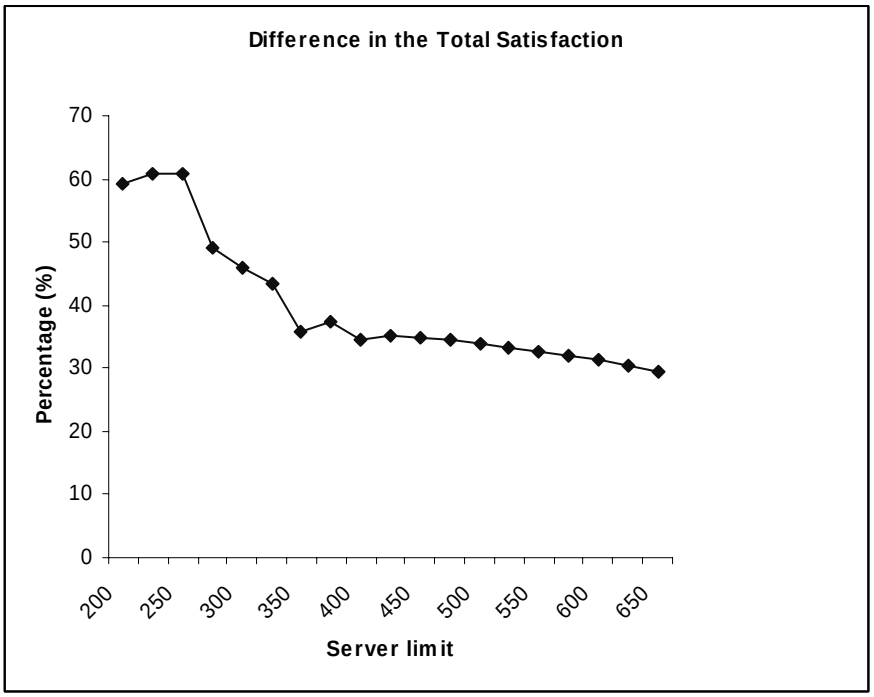

Figure 28. Difference in the total satisfaction between the OCoV heuristic algorithm and the optimal algorithm.

Figure 27 summarizes the comparison results for the running time of the OCoV heuristic algorithm and the optimal algorithm. The graph shows clearly a major reduction (in the order of 10^6) in the running time of the algorithm to find the best combination of streams and their QoS configuration parameters. This reduction in running time definitely improves the response time of the data source to deliver customized content. This reduction came at a cost of (30-60%) reduction in the total achievable user satisfaction, as Figure 28 shows. It would be interesting to optimize the achievable total user satisfaction of the OCoV heuristic algorithm, and still keep the same running time. We will leave this issue for future work.

5.4 Conclusion

In this chapter, we have proposed an end-to-end rate-based mechanism for the selection of the optimum QoS for media broadcast that relies on the knowledge of the user preferences and bandwidth limitations of all receivers. Based on this information, the source will select the number of QoS variants for the given media stream and the QoS parameters for each of these variants. To limit the problem of feedback implosion from

the receivers, we use a virtual representative for all receivers within a given group of receivers. We have considered several algorithms to determine the QoS preferences of the group representatives. We have also presented a heuristic algorithm to computer the number of streams as well as the QoS configuration parameters for each stream, and compared its performance to the optimal algorithm.

Chapter 6

Architectures for Personal Mobility Systems

6.1 Introduction

A natural consequence for the multiplicity of devices and access networks is to try to integrate seamlessly similar services across networks, and to enable users from a certain network to use services in other networks. Users that subscribe to services in one network would like to have the flexibility of accessing these services through devices connected to other types of access networks. Examples such as accessing the e-mail inbox from a regular telephone line, or accessing messages in a voice mailbox through the Internet are typical kinds of accessing services from outside the native networks

The idea of “always connected” has always been a driving vision of the key players in the telecommunication world. This vision is beneficial for users who are on the move in order to provide them with constant access to telecommunication services. This vision has also created several forms of mobility, such as device mobility, session mobility, and user mobility. Device (or terminal) mobility is defined as the ability of a device to move from one location to the other without losing network connectivity. This form of mobility is achieved by technologies at the network layer such as cellular telephone network, or Mobile IP [130]. Session mobility is mobility at the application layer, where technologies like Hot Desk [131] allow users to stop a session at a certain terminal, and to continue the session at another terminal from the same point where it was stopped. User mobility, also known as personal mobility, is defined as “the ability of end users to originate and receive calls and access user-subscribed telecommunication services on any terminal in any location, and the ability of the network to identify end users as they move”[132]. While device mobility has been the focus of research for many years, the area of user and session mobility are still in early stages.

The first notion of personal mobility started with the Universal Personal Telecommunication (UPT) in the ITU Telecommunication Standardization Sector, which was defined as the ability of a user to receive telecommunication services anywhere,

anytime and through any telephony terminal. Each user in the UPT paradigm is assigned a unique personal UPT number and a personalized service profile that serves as an application level routing algorithm. UPT was originally produced for the PSTN and it supports only telephone terminals. Extending the same concept to support different types of terminals and networks has lead to what is currently known as personal mobility.

Several research groups proposed architectures to solve the problem of personal mobility. Notable among these architectures are the Mobile People Architecture (MPA), the Internet-core Network Architecture for Integrated Communications (ICEBERG), the Telephony Over Packet networkS (TOPS), Seamless Personal Information Networking (SPIN), Call Management Agent (CMA) System, the Session Initiation Protocol (SIP) and finally the Seamless Personal Information Networking (SPIN). In this chapter, we will present several architectures that were proposed as a solution for the personal mobility problem. But before we talk about each of these architectures, we will define clearly the problem of service mobility.

6.2 Personal mobility: Problem statement

The wide spread and diversity of communication devices that a person has access to made the task of reaching that person a big challenge. A caller always has to guess when, where, and how to reach a certain callee. Managing and configuring communication user preference as to when and who have the permission to reach the user on each device is also a big task for the user. In addition, keeping the user always connected without jeopardizing his/her privacy is always a main concern. Following is a description of the many challenges and issues related to device multiplicity and personal mobility:

- **Addressing people rather than communication devices**: Until today, people still perform the conversion in their mind between persons and the identification of their communication devices. In addition, people use application or device specific address even when the target entity is a person. The phone directory is just one service to help alleviate the problem, though the user still has to search manually for the phone number, and it works only for telephones. In addition, there is currently no one directory that includes the user's phone number, cellular phone number, SIP

URL, Skype [133] user id, email address, and other device addresses which are accessible by different telecommunication technologies.

- **Locating the called party**: Since personal mobility is all about reaching a person, finding the workstation, the phone, the cellular phone, and in general, the device that the user currently has access to is a major task. Industry estimates quote that only ten percent of phone call results in productive conversation: most of the calls ends up reaching a voice mail, go unanswered, or reach the person at an inconvenient time [134]. This is only when using normal telephony and when each user has one phone number; having multiple numbers makes this problem worse, and the worst is when considering other candidate ways of communication such as trying to reach the person on his cellular phone, pager, or using VoIP through his workstation.
- **Customizing communication**: Most people would like to customize their availability on each device according to different criteria, such as the time of the day, the day of the week, the type and format of received content, and even according to different roles of the caller and/or the callee. Connecting with a person according to his preference makes the communication session desirable and un-intrusive. While traditional applications and services do not offer much choice for the user (only ON/OFF option)(we refer to this as **low level** customization), new Internet based telephony application such as IP-telephony allow their users to fine tune a number of parameters. A user may customize several options for the applications, including customizing the type of media and the quality for each media type (we refer to this as **high level** customization). For instance, an employee may prefer using only telephone-quality, voice communication when talking with a co-worker, while insist on CD-audio quality when talking to his manager.
- **Acquiring new devices**: Still another problem is with acquiring new devices. When a user subscribes to a new service, he/she has to pass the address of his/her new service to other people before he/she can start receiving calls through this service. For instance, a person that has subscribed to a cellular service, might not be reached by a second party if the second party is not aware of the access number of the service. Publishing the address for a new service is usually an awkward and lasting

task. A user would prefer an environment similar to a “plug and play” where new subscribed services can be used the moment they are plugged into the environment. In such an environment, the user does not have to worry about passing around the address of the service, and the environment takes care of routing incoming calls to the service without requiring the caller to be aware of the address of the service. De-activating a telecommunication service has the opposite effect on person reachability.

- **Using public services near the user:** While the set of services available for the user changes when the user acquires a new service, it can also be more convenient if the set is updated automatically with additional public telecommunication services that the user may transiently get access to. For instance, it would be convenient for a user walking into a common room to be able to receive communication calls on the phone in that room. Changes in the set of services available to the user to include such services should be done transparently to the user who might be un-aware of the possible available services and their addresses.
- **Maintaining the user’s privacy**: Most users agree that providing customized ubiquitous access should not jeopardize their privacy. While certain access networks, such as the wireless network can conceal the location of the user, wired devices such as telephones and computers can easily reveal the location of their user. Information about which device is used by the called party and hence his location should be hidden from the caller. A similar logic applies also for the caller, where the called party should in no way be able to detect the address of the caller’s device and hence his location. While location privacy might not be an issue for some users, other users might be ready to pay extra fee to keep their location private. Another important privacy threat can be directed at the personal preferences of the users. Most users would definitely prefer to hide their preferences for communications, as this information can be considered as private information. For instance, having a blocking list public would be an embarrassment to most people.
- **Data trans-coding and protocol bridging (applications inter-operability):** personal mobility is tied to application independent communications. Connecting to any of the mobile user’s devices from a non-compatible device that uses different

protocols for signaling and data transfer necessitates the existence of one or more gateway/proxy between communicating parties. A PSTN-Internet gateway for instance has one interface connected to the PSTN and another interface connected to the Internet. A connection from the Internet to the PSTN must go through a gateway that works as a terminating end-point for both devices on the Internet and the PSTN. Devices with incompatible codec require intermediate proxies that can trans-code formats to allow data flow between the two devices.

6.3 Architectures for personal mobility

In this section, we will present a number of architectures for personal mobility. The next section lists a number of deficiencies with these architectures. Chapter 6 presents our Mobile Internet Telecommunication (MobInTel) architecture that addresses these issues.

6.3.1 The Mobile People Architecture (MPA)

The Mobile People Architecture (MPA) [135] proposed by the **Moquitonet** group at Stanford university aimed at addressing the deficiency of the network protocol stack, where the "personal" factor in telecommunication is absent. The architecture presents an extension to the current OSI 7-layer model, to change the end-point communicating from applications to persons. It also routes the communication to the person, regardless of the person's location or his current application.

The MPA architecture proposes a *person layer* that bridges the gap between the communicating users and the underlying communication infrastructure. Within the *person layer*, each user has a *personal proxy* that acts on his/her behalf; it works as a personal level router that tracks the location of the user, accepts, converts, and routes communication calls based on the type of the communication request and the user preferences. Each *personal proxy* has an address that is called the Personal Online Identifier (POID).

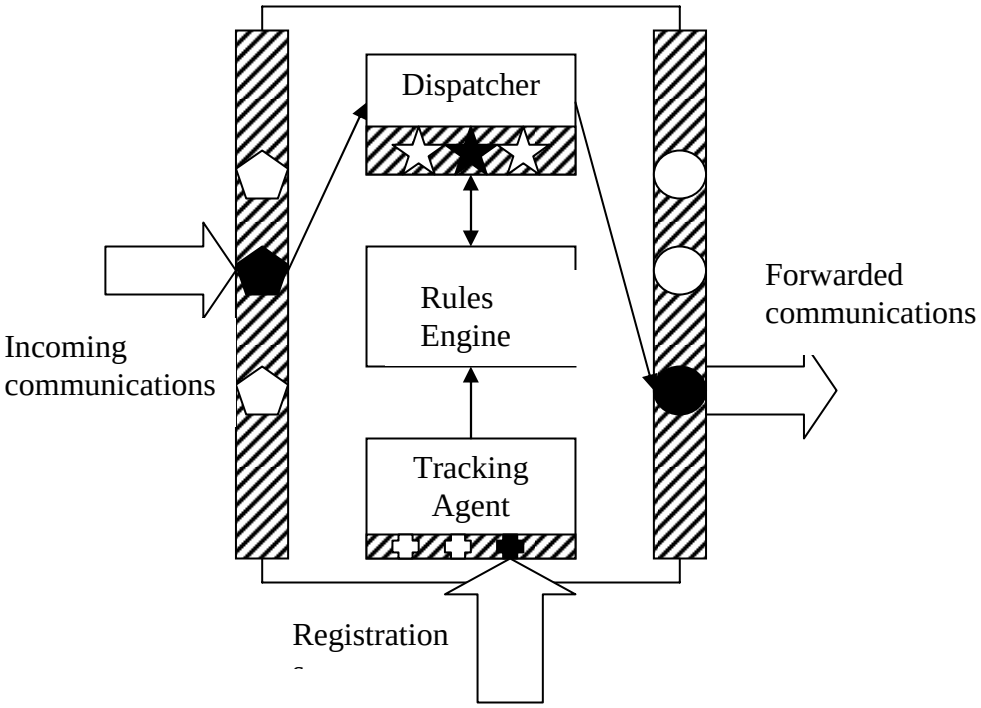

Figure 29: Architecture of the MPA *personal proxy*.

The *personal proxy* has four components, as shown in Figure 29:

- **Tracking Agent**: The Tracking Agent keeps track of the location or application where the user is most likely to be accessible. Applications can be registered with the tracking agent manually, automatically or according to a user-specified profile.
- **Rules Engine**: Based on information from the Tracking Agent and user's preferences, the **Rules Engine** directs the dispatcher on how to forward the communication request. The **Rules Engine** specifies also the description of the desired output format.
- **Dispatcher**: when the *personal proxy* receives an incoming call, the **Dispatcher** asks the Rules Engine for instructions on how to route the incoming call. Based on directions from the **Rules Engine**, the **Dispatcher** instantiates the right application drivers that trans-code the data and control signals to the appropriate application-specific format.

- The **Application Drivers** act as adapters between non-compatible applications. Four types of drivers have been proposed: Session Drivers that receive and send communication requests, Messaging or Protocol Drivers that parse the communication requests/response and generate the corresponding meta-data, Content Drivers that search for a certain pattern into the data content, and finally Conversion Drivers that convert data from one format to another.

During session initiation, a caller passes the POID of the callee to the local *personal proxy*, which queries then a directory service such as LDAP for the Proxy Application-Specific Address (PASA) of the callee's personal proxy. The *personal proxy* provides the directory service with the POID of the callee and the application type the caller is willing to use. The directory service returns the Proxy Application-Specific Address (PASA) of the callee's personal proxy. Each *personal proxy* has an application address for each application of the user, which is different from the application's address. The invitation is then sent to the callee's *personal proxy*, which in turn forwards it to the application specific address (ASA). Only the ASA of the *personal proxy* are included in the directory service, while the real ASA's of the applications are kept secret with the *personal proxy*, protecting hence the privacy of the callee. If the user does not care about his/her privacy, the ASA can then be included in the directory and returned to the querying application. The *personal proxy* would then be responsible only for updating the ASA in the directory service, without interfering in the initiation of the session. In addition, outgoing requests also protect the caller's privacy.

The MPA architecture provides full privacy for both caller and callee using the *personal proxy*. The use of POID allows for personal level communication, independent from the type of application. The Tracking Agent allows for the mobility of the user between several applications, though it does not cover for dynamic discovery of devices in ad-hoc environments such as ubiquitous environments. The architecture allows for customizing the user's communication, though the level of customization is not specified in the published papers. In other words, it is not known whether the architecture allows the user to specify the quality for the communication session or it just allows the user to

specify only the device to use for the session (low level customization). The architecture allows also for application interoperability by using application drivers.

6.3.2 ICEBERG: Internet-core Network Architecture for Integrated Communications

The ICEBERG [136] architecture is an Internet-core based architecture that integrates together telephony services over different networks. It uses the Internet as the base infrastructure for connecting devices with different access technologies. A number of key components are introduced in the ICEBERG architecture:

- **Clearing House:** The Clearing House is the control and management entity for the traffic flow between the ICEBERG network plane (and hence the various access networks) and the Internet Service Providers (ISP's). It acts as a bandwidth broker in establishing data paths that meet certain QoS requirements. It also performs authentication, authorization and accounting for ICEBERG users.
- **Name Mapping Service:** The Name Mapping Service acts as the directory service in the ICEBERG architecture. It maps an ICEBERG Unique ID (iUID) associated with each user to a set of communication devices.
- **Preference Registry:** The Preference Registry is a service that stores and processes the user's preferences. Given the caller's and callee's iUID's, the time of the day, and some other dynamic information such as the current location of the user, the Preference Registry finds the device that is best suitable for the requested communication session. Users use the Preference Registry to express and customize their communication preferences.
- **Personal Activity Coordinator:** The Personal Activity Coordinator (PAC) acts as a tracking agent for the end-user. It keeps track of the user's current location, current activity and any other dynamic information that could be used during the session setup. Information in the PAC is used by the Preference Registry to provide more accurate information about the current activity of the user.
- **ICEBERG Access Points (IAP):** An ICEBERG Access Points (IAP) is a gateway that connects the ICEBERG network to an access network such as the PSTN, GSM cellular network, or the pager network. It is access network

dependant, and it is responsible for converting control messages and data flow between the networks with different data and control protocols.

- **Automatic Path Creation Service:** The Automatic Path Creation Service (APC) is responsible for the computation and establishment of data flows between the end-points. Depending on a number of constraints, such as available network resources, the APC might add several trans-coders along the data path between the end-point devices.
- **Call Agent:** The Call Agent (CA) is a process that is responsible for setting up, maintaining, and tearing down communication sessions. It uses the Clearing House for authentication, accounting and resource reservation. It interacts with the Name Mapping Service and the Preference Registry to map an iUID into the device to use, and with the Path Creation Service for establishing data flows. Each user has one Call Agent per device, per session. Call Agents are created by the iPOP when a connection request is received or initiated.

All the components of the ICEBERG architecture reside in the ICEBERG network plane (Figure 30) that connects several access networks over the Internet. ICEBERG Points of Presence (iPOPs) are connected to all access networks through ICEBERG Access Points (IAP); they are also connected to a Clearing House and to Internet Service Providers (ISP).

While most of ICEBERG architecture components are very common to several personal mobility architectures, ICEBERG uses a soft state signaling protocol to setup and maintain the session state. All call agents in one communication session share their state information over one multicast session. Periodic call state information is sent from the IAP to the iPOP. The iPOP passes this state information to the corresponding CA, which in turn propagates this information to other CA’s on the multicast channel. This call signaling mechanism makes ICEBERG a very good platform for simple and quick service creation and deployment.

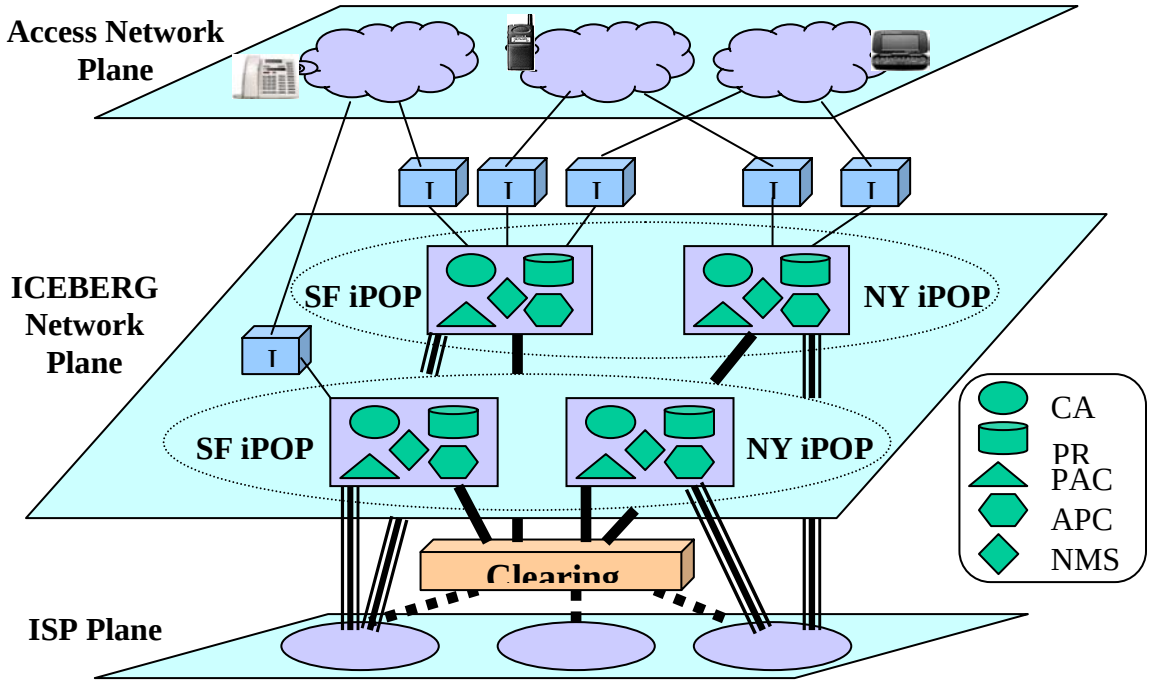

Figure 30: The ICEBERG Architecture (from [136]).

ICEBERG shares many of the MPA properties in the support for personal mobility. ICEBERG provides customized communication through the Preference Registry. User's location privacy is maintained by returning the address of iPOP to the caller, rather than the address of the device to use. Interoperability between different access networks is provided by using different IAP's. The ICEBERG architecture suffers though from the same drawbacks of the MPA, in that it does not support device discovery and high-level communication customization.

6.3.3 Telephony Over Packet networkS (TOPS)

The Telephony over Packet Networks (TOPS) [137] architecture was designed to exploit the flexibility of the packet network, and the increased intelligence of the end devices. TOPS users may have several devices, and they can control how communications are directed to these devices.

The major driver for the TOPS architecture was to extend the traditional directory service to support user mobility and location management. A directory service in the

architecture allows users to register their devices as well as the selection directive or profile to handle all communication calls. Each user, identified by a ***distinguishing name***, keeps a user record (Figure 31) that contains his/her personal information and profile in the directory. Using the ***distinguishing name***, TOPS users may contact one another without having to first figure out each other's phone numbers or email address.

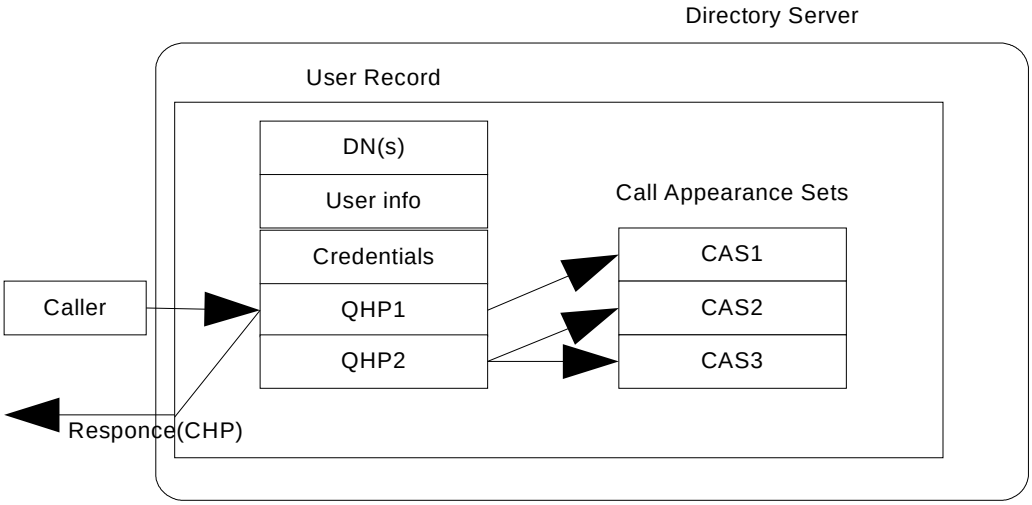

Figure 31: Structure of a user record

There are three main components in the TOPS architecture:

- **Directory Service:** The directory service in the TOPS architecture is a convenient way to protect the user from having to deal with low-level addressing schemes and to permit personal mobility. The directory service maps a distinguishing name of a user to a set of call appearances, based the conditions and preferences in the call handling profile. Each call appearance holds enough information about one terminal where the user can be reached. The directory server allows the user to customize the query handling and call setting on a call-by-call basis, depending on the caller, the time of the call, the urgency of the call, and some other conditions. Using the user query handling profile, the directory service gives the user a full control over which call appearance is returned to the caller.
- **Application Layer Signaling (ALS):** Applications running on top of the TOPS architecture use the Application Layer Signaling (ALS) protocol for establishing and

controlling the communication session, negotiating capabilities, and requesting advanced services. The ALS protocol allows the application to add and remove media channels. The two basic messages of the ALS are the INVITE and RESPONSE messages; the INVITE message is sent over the default Logical Channel (LC) of the *call appearance* returned from the directory query. The INVITE message contains the attributes of the logical channels as desired by the caller. The caller may also take into consideration the capabilities of the callee terminal(s) before sending the INVITE message. The RESPONSE message describes which of the requested channels are available, as well as any additional information provided by the callee. After the caller's application receives the RESPONSE message, it can proceed with the resource reservation, and when all the requested resources are made available, it sends a RING message to the callee's application which answers with an ANSWER message, containing the set of media the callee actually decided to accept.

- **Logical Channels:** The Logical Channels (LC) layer provides the application layer with an abstract uniform communication interface, independent of the underlying network. Several LC's with similar QoS transport requirements can be multiplexed onto one transport channel.

The TOPS architecture provides also a mechanism to support both loosely and tightly coupled conference control mechanisms. For tightly coupled conferences, the TOPS architecture provides conference controllers that perform control functions for conferences including capability negotiation and policy access to the conference. The ALS protocol messages, CONFIGURE, CNF_ACK, MODIFY, MOD_ACK, END, END_ACK are used to create, modify and terminate a conference. TOPS allows also both centralized and decentralized mixing schemes for media streams.

Similar to the MPA architecture, the TOPS architecture allows for personal mobility using the directory service. Personal level addressing is achieved by using a *distinguished name*, which the directory service maps into a terminal address. The Query Handling profiles allows for user customization of communication, though like the MPA architecture, it allows only for low-level customization. TOPS supports only terminal mobility, but not dynamic discovery of devices and services. TOPS also does not support

application interoperability. Location privacy is not provided in TOPS, since the Query Handling Profile returns the call appearance of the device of the user, which could be used to deduce the location of the user.

6.3.4 Call Management Agent System

The Call Management Agent system (CMA) [138] is an agent-based architecture proposed by the Voice-over-IP Consortium for managing multimedia communication sessions. In the architecture, each Call Management Agent (CMA) represents a person or a group of persons and manages the communication access to their devices. Each CMA has the logic of the user or group it represents, which allows it to select the communication device for the communication session.

Each CMA is identified by a CMA Name (email-like address), which is used to locate the CMA server where the CMA resides. Since a CMA server may host one or many CMA's, the CMA Name is also used to identify the intended CMA on the CMA server. The system assumes the existence of DNS-like directory for locating the CMA server of a certain CMA.

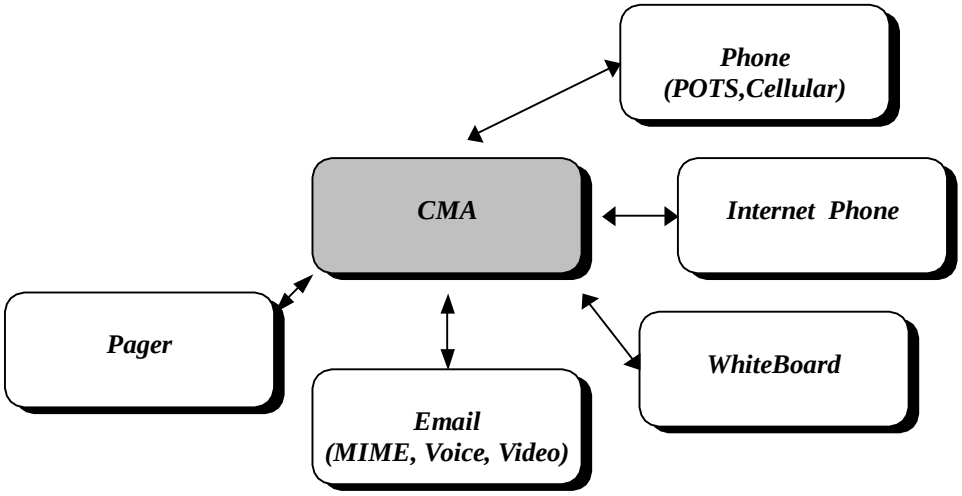

Figure 32. A CMA manages multiple communication terminals

Each user's device in the CMA architecture has a Communication Terminal Specification (CTSpec) that holds all the information about the device. The CTSpecs are stored as properties of the CMA of the user. A query to the CMA of a certain user will return a list of CTSpecs representing the set of devices that could be used to access the user at this point of time.

The CMA system defines also a protocol, called the CMA Protocol (CMAP), which is used for communicating with the CMA server. Using the protocol, the caller's CMA can send a **RESOLVE** request to the callee's CMA server, containing the callee CMA Name, and possibly the caller CMA Name, the CTSpecs for the caller's own devices. The RESPONSE message would include a list of CTSpecs for the callee's devices, or a RIDERECT response, which instructs the caller CMA to redirect the RESOLVE request to another CMA server. Other CMAP messages include: GET message, to retrieve certain properties of the CMA, **CREATE** message, to add some properties to the CMA, and **CHANGE** message, to change some already existing properties of the CMA.

The CMA architecture shares with the TOPS architecture a number of properties for personal mobility, including personal level addressing, customization of communication, lack for location privacy, and lack of support for application interoperability. They differ only in that the CMA architecture defines its own protocol to access the directory server (CMA server) while the TOPS architecture uses any directory access protocol. The CMA architecture also does not define a signaling protocol like the ALS protocol of the TOPS architecture.

6.3.5 Session Initiation Protocol

In Section 3.5.2, we have talked about the Session Initiation Protocol (SIP) [49] as a VoIP signaling protocol. SIP integrates both VoIP location management and the application layer signaling into one protocol, making it a good candidate for many applications that requires personal mobility. In this section, we will discuss how SIP can support personal mobility.

The SIP protocol, as we described it earlier, uses an addressing scheme similar to e-mail addressing to contact a person, regardless of the address of the device. The mapping between the user name and the device to use is done on the intermediate servers.

SIP defines a number of logical entities: user agent, redirect server, proxy server and registrar server. The user agent originates and terminates SIP messages. The redirect server receives requests from user and responds with a message that includes the location(s) where the targeted user agent should be contacted. The proxy server forwards request messages to the targeted user agent or another proxy server. User agents use the service of the registrar server to register their current location. The registrar server provides user agent location information through a location service. Proxy servers use these location services to forward the incoming, while location information sent to a redirect server is returned to the calling user agent.

A SIP proxy or redirect server and a registrar server can be used to support personal mobility. A user might run a user agent on each of his devices, and each user agent would register its addresses with the registrar. An incoming request might be diverted to any of the user agents based on certain call processing rules: the user can use either the Call Processing Language (CPL) or the SIP CGI to express his/her processing rules. Example of personal mobility support in SIP is illustrated in Figure 33[139]. The example shows a user *Alice* who wants to be reachable via a PSTN phone, a wireless device and a PC. User A*lice* would register these devices with the SIP server at “yahoo.com” and “comlumbia.edu” together with the rules to handle incoming calls, and just publish her addresses(alice17@yahoo.com, alice@columbia.edu, 7000@columbia.edu, and Alice.McBeal@columbia.edu). A call to any of *Alice*’s published address would be forwarded to any of *Alice*’s devices, based on her preferences

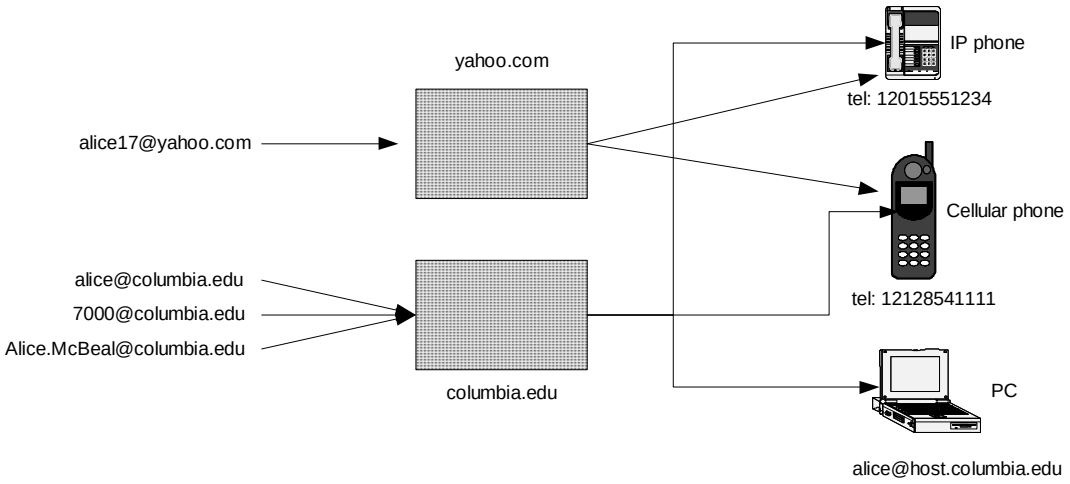

Figure 33. Example of personal mobility support in SIP

The SIP architecture has a number of shortcomings when it comes to personal mobility: first of all, all the user devices and applications must be registered directly with the SIP registrar to become available for the user. Second, SIP does not have any dynamic discovery mechanism for devices and applications. Interoperability and location privacy were not included in the protocol. But as we mentioned earlier, a SIP user can use the SIP CGI or CPL to create a script that is executed by the SIP server when the user receives or sends a communication request. This script usually contains the user customization for his communication sessions.

6.3.6 Seamless Personal Information Networking (SPIN)

The Seamless Personal Information Networking (SPIN) [9] project at the National Research Council of Canada (NRC) in Ottawa introduced a seamless messaging system for the management of personal incoming messages across heterogeneous networks. The system allows the user to customize the way the system handles the messages in his/her multimedia mail-box.

The system architecture pf SPIN is composed of four major components: Message Watcher, Personal Agent, Service Process, Device Manager, and Service Process. The complete architecture of the system is showed in Figure 34.

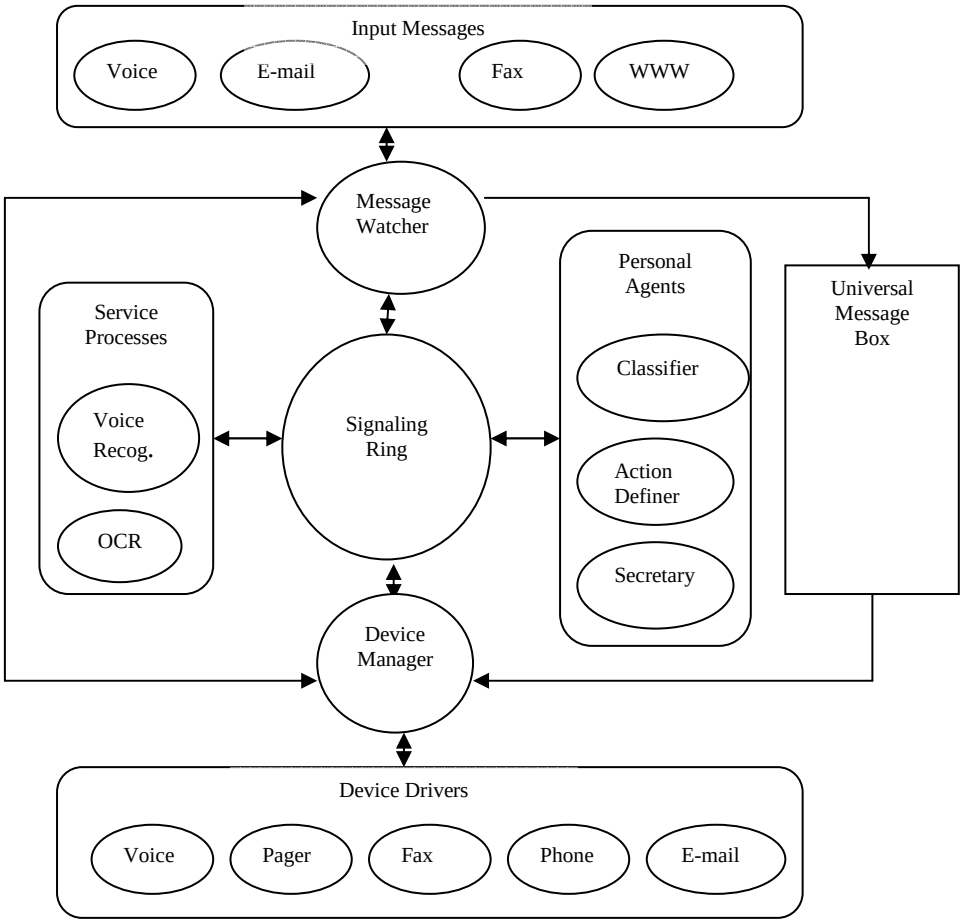

Figure 34: SPIN system architecture

The Message Watcher awaits incoming messages and formats them in a form that the system can process. The Device Managers are device-dependant drivers and are responsible for delivering the message to the specified user device. The Personal Agent has three main sub-components: the Classifier, the Action Definer, and the Secretary. The Classifier, triggered by the Message Watcher, is responsible for classifying the message based on the rules provided by the user. After the message has been classified, the Action Definer is triggered to define the actions to be applied on the message. These actions are interpreted by the Secretary. The Secretary might require additional resources to perform certain actions request by the Action Definer. For instance, the Secretary might require the service of a tracking agent to deliver urgent messages to the user. Both, the Message Watcher and the Device Manager use the Service Process: the Message watcher triggers the Service Process to convert incoming messages into the system's unified format for

message processing, and the Device Process uses the Service Process to convert a message in the system's unified format to the format acceptable by the end-device.

The SPIN architecture does not use any addressing scheme, and the caller still has to know all the addresses of the callee. It hides though the information about the device that received the message or call. Additionally, the architecture allows the user to specify the rules for handling calls, and hence provides a partial personal mobility service. The system does not support though high-level customization or dynamic discovery of devices.

6.3.7 Personal mobility in telecommunication networks.

As we have mentioned earlier, the first notion of personal mobility started with the Universal Personal Telecommunication (UPT) in the ITU Telecommunication Standardization Sector, which was defined as the ability of a user to receive telecommunication services anywhere, anytime and through any telephony terminal. UPT is an example of personalized service that the POTN can provide. UPT is just one manifestation of the desire of the telecom providers to provide their customers with much more customized services. Telecom providers soon realized that providing customized service with their traditional systems can be a very daunting task (especially with many feature interaction problems) which calls for drastic changes in the architecture of these communication systems.

A number of researchers [140, 141, 142, 143] proposed the use of Multi Agent Systems (MAS) as an architecture platform for next generation telecom services with minimum feature interaction. In [143], the authors presented an architectural framework that identifies the different components of a distributed telecommunication system; the architecture used different agents to represent different components, "keeping their concerns and responsibilities separate and well-defined" [143]. Three types of agents were identified in the architecture: (1) user agents, enforcing user profiles and preferences, (2) service agents, representing with different logic representing different telecommunication services, and (3) resource agents, that manage the functionality contained in resources. The processing model of the architecture requires that the user agent uses a set of dispatch rules to determine when to send a message to other user,

service, or resources agents. The dispatch rules provide by the services agent to which the user has subscribed. Some dispatch rules are also created from the user's profile. This processing model enables the user agent to make decisions that resolve interactions independently of the specific services to which the user has subscribed. In [140], users' preferences and constraints were expressed as features or policies; agents representing users use negotiation mechanisms to solve conflicts between users' policies. During the negotiation process, the user agent uses a local tree structured goal hierarchy that represents the user's policy to generate proposals, determine acceptability of received proposals, and generate counter-proposal to proposals received from other user agents. This negotiation process between different agents enforcing the constraints of different users provides an automated method for detecting and resolving conflicts at run time.

Recently, the concept of Virtual Home Environment (VHE) was introduced with the introduction of third generation wireless mobile networks like the Universal Mobile Telecommunications System (UMTS) [144,145]. The VHE was defined as an environment that enables the user to receive customized and personalized services, regardless of location, access network or terminal type. VHE gives the user the ability to roam between different telecommunication networks and still receive the same set of services, thus giving the "feeling" of being on the home network. In addition to seamless access to subscribed services, the VHE provides also personalization of services to customers through adaptive terminals. A number of MAS [146,147,148] were also proposed for the provision and support of the VHE. Our architecture goes beyond the VHE concept, which focuses mainly on Intelligent Network (IN) services. Our architecture focuses on services that are available to a user in a certain context, without a prior subscription to these services.

6.4 What is still missing in personal mobility?

All the previous architectures, except SPIN, support the use of a naming scheme that allows users to be addressed independently of the addressing scheme of their devices. Each user is assigned a unique identifier (id), and the architecture takes care of mapping between the user id and the address of his/her devices. Using such a naming scheme allows addressing people rather than communication devices. For the caller, it means

only remembering the one unique id is sufficient to locate the called party on any of his devices, while giving the callee the power to manage and control their availability through their devices.

Each of the presented architectures allows its users to customize their communication sessions according to their preferences, as to when, where, and from whom to receive calls. While all the architectures include services for routing calls based on user specified preferences, none of these architectures performs routing decision based on the combination of the user's high-level preferences. Chapter 7 describes how the user's satisfaction is used to manage communications. Moreover, only the SPIN and ICEBERG architectures talked about using intermediate trans-coding services to support type mismatch between different communicating devices. The SPIN approach is a primitive one, while the ICEBERG approach is based solely on low level QoS parameters.

In additions, none of these architectures included dynamic discovery of public services in environments like ubiquitous computing environments. Factoring in these services in the surrounding of the user adds more dimension to personal mobility. Chapter 7 describes an architecture to support personal mobility in ubiquitous computing environments.

Chapter 7

MobInTel: Mobile Internet Telecommunication architecture

7.1 Introduction

Electronic communication has proliferated deeply into our daily life, and it is becoming more than normal to have several email accounts, a telephone at home and at work, a cellular phone, a workstation at home and in the office, a laptop with a wireless connection, a pager, and the list can go on and on. While each individual enabling communication technology is advanced and stable, a framework for enabling the management of a collection of these devices to meet the owner expectations still does not exist. With multiple access points, it is becoming a challenge to connect to the person at the right device, right time, and using the right configuration parameters. It is also more challenging for the user to decide on which device to use to make a call to somebody, since the user has to worry about the quality of the call, and the price he is charged to make the call.

Additionally, to provide authentication, authorization, billing for each of these services, all service providers use central or distributed proprietary data-base servers in their domains. With each service provider being interested in the profile of the user from the point of view of the service it provides, information about a single user ended up being stored in separate independent location. A complete comprehensive user profile that controls the use and preferences of a user to all the devices and services he/she subscribed to is still missing.

User mobility allows a user who has multiple devices to control how communication session are established through these devices. While device mobility has been the focus of research for many years, the area of user and session mobility is still in early stages. In Chapter 6, we have presented a number of pilot projects [135,136,137,138,49,9] that have focused on user mobility, but none of these architectures has considered multimedia Quality of Service (QoS) requirements and

parameters. Additionally, while all these architectures provide solutions to user mobility, most of the selection criteria are based on the time of the day, the availability of the device, and the role of the user. None of the selection criteria is based on user satisfaction and multimedia QoS preferences and device capabilities.

In this chapter, we address the issue of user and service mobility and show how our framework presented in Chapter 4, can support them. The chapter is divided into two parts: in the first part (Section 7.2 – Section 7.5), we present our Mobile Internet Telecommunication (MobInTel) architecture which is based on the concept of having complete comprehensive information about the user stored in a single user profile at a directory server called the Home Directory (HD) of the user. In addition to the information about the devices the user owns, the user profile contains the set of policies to control incoming and outgoing communication requests for the user. The profile includes also the ranges of acceptable values for each QoS configuration parameter. Our MobInTel architecture treats the user's satisfaction with the quality of service of the multimedia session as an essential factor to select among multiple available services. With this logic stored with the data, end users take the power to control the logic and access permission to their devices according to their own preferences. The Home Directory Agent (HDA) uses the framework presented in Chapter 4 every time the user receives an incoming call through the HD. The HDA can be also invoked by the user in order to find (and establish) the best way to make an outgoing communication call.

In the second part of the chapter (Section 7.6 – Section 7.12), we present an extension of the MobInTel architecture to support personal and service mobility in ubiquitous computing environments. The extended architecture leverages technologies in short-range wireless communication, such as Bluetooth, to construct the WPAN; it also leverages service discovery protocols, such as Jini, SDP, SLP, and Salutation, to discover services available just in the WPAN of the roaming user. The extended architecture supports also optional service mobility, which allows, when possible, the service currently in use to follow the user as he moves from one location to the other. An extended instance of the HDA, called the Personal Agent (PA), runs on a personal device carried by the user; the PA triggers the service discovery, service selection and controls

service mobility. The PA also enforces the user's policies and preferences stated in the user profile.

The rest of the chapter is organized as follows: Section 7.2 describes in details the content and operations provided by the HD, and its controller, the Home Directory Agent (HDA). Section 7.3 presents the architecture of the HDA. The automatic QoS parameter selection process is discussed in Section 7.4. The HDA support for session establishment is explained in Section 7.5. In Sections 7.6 and 7.7, we start our discussion on extending MobIntel to ubiquitous computing environments with a literature review of a number of architectures for personal mobility with highlights to their limitations in ubiquitous computing environments. We then propose our extension for the MobInTel architecture for supporting personal mobility in ubiquitous environment and its main components in Section 7.8. Section 7.9 shows how our architecture supports service mobility. Section 7.10 continues with more details about the usage scenario introduced in Section 7.6. Our prototype and performance measurements are presented in Section 7.11. We finally conclude in Section 7.12.

7.2 Home Directory

In the following sub-sections, we will focus on different aspects of the HD, including the content of the HD, subscription to the HD, updating the HD, benefits of the HD, and finally give an example to show how the architecture supports multimedia session establishments.

7.2.1 Content of the Home Directory

The Home Directory acts as a storage place for user-centric profiles. Each profile captures the personal properties and preferences of the user it represents. Properties include, but are not restricted to personal information such as name, and employer, list of devices and services controlled by the user, as well as the access permission list to all these devices and services. As for preferences, they can cover the parameters for all applications running on the user's devices, especially the user's application level preferences concerning the quality of service for the multimedia applications. These preferences can be associated with groups or persons, and they cover the choices

concerning the receiving/sending of audio and video QoS parameters, such as the frame rate, the resolution, the audio quality and the importance (weight) of different media types. Users express their preferences by specifying the minimum and the ideal acceptable value for each QoS parameter, and possibly a function that maps between the QoS parameters and satisfaction values as explained in Chapter 4. A section for public preferences may also be included, which could be used to specify that the user is interested in news about the stock market. This section, which we call *public preferences* section, can be used by content providers to provide tailored advertisements and services. An example of a typical user profile is shown in Figure 35. An XML schema for the user profile is shown in Appendix A.

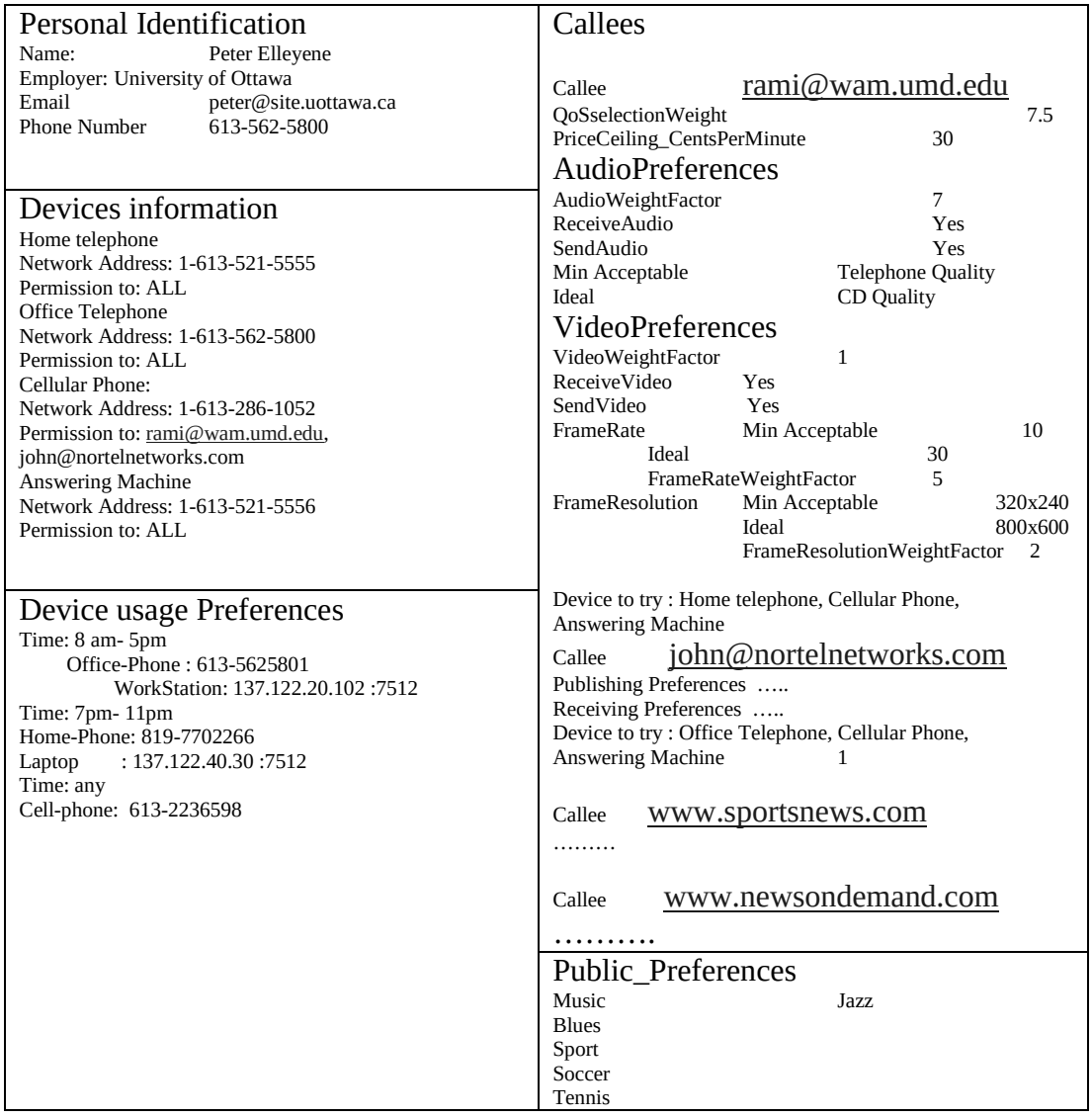

Figure 35: User Profile

7.2.2 Subscription to the HD

In order to benefit from the system, each user is required to create a profile on a directory server; the directory server be called then the Home Directory of the user. All other directory servers would be referred to as Foreign Directories. Agents managing these directories are referred to as Home Directory Agent (HDA) and Foreign Directory Agent (FDA), respectively.

When a user subscribes to a domain, he/she completes all the basic information in the profile. The HD can provide a template profile, and users would add more information into their profiles as they acquire new devices or change their preferences. Security keys for authorization and authentication purposes are also assigned at this point, though they can also be updated later. The user will then be assigned a universal distinguishable identifier that would then be used when somebody needs to contact him. In our prototype implementation, we have adopted the SIP naming scheme, and used the well established SIP protocol as the main signaling protocol in the system.

7.2.3 Updating the content of the HD

Even though the initial version of the user profile will contain little information about the user and his devices, modifications to this profile will be done as the user gets acquainted with the system or when the user acquires new devices. Modifications to the profile can be done either manually or dynamically. Manual modification may include using a graphical interface to the HD. For dynamic modification, the HDA can monitor the behavior of the user, and extract some common patterns and use them to update the user's profile. This functionality will tune the user profile to best fit the user's needs.

As another extension to this dynamic update of the user's profile, we have incorporated the Bluetooth [149] technology into the system, as we shall see in Chapter 7. Information about Bluetooth devices detected by a handheld Bluetooth device can be added dynamically to the user profile. The set of detected devices might include personally owned devices as well as public services and devices that the user has permission to use. In this case, the system would be more dynamic and accurate in locating the user. Chapter 7 gives a detailed description of how the architecture was

extended to support dynamic update of user profiles, for users roaming in ubiquitous computing environments.

7.2.4 Benefits of the HD

The HD architecture has a number of advantages. Besides shielding the end-user from the problems of diversity of network technologies, protocols and end-system devices, and dynamic addresses, it also has several other benefits. Below is a list of direct benefits to its users.

- User centric communication architecture: Most of the current addressed entities in a communication session represent just endpoint devices, never a user. Even for a person-to-person communication session, it is always the phone number or computer address that is dialed in, and the end-user is never addressed himself. The MobInTel architecture adopted the SIP addressing scheme, where each user is assigned a unique identifier which forms an umbrella for all the devices and services the user has access to. Each user will be addressed with his/her unique identifier, independent of the device used for communication.
- Management of user devices according to the user preferences: The MobInTel architecture allows the end-user, as ultimate holder of many devices, to control how these devices should handle, coordinate and perform up to the maximum possible satisfaction of their owner. The architecture allows the end-user to specify which device to use depending on various criteria such as the caller, the time of the day, the importance of the call, or the current role of the caller and callee.
- Service creation environment: one of the most important benefits of the MobInTel architecture is that it provides a rich environment for creating and running user customized services. Services like *call-forward, call blocking, follow-me* can be easily created using the user profile. These services can be built and introduced by the user himself in no time. New complicated services such as the ability to record a conversation during a communication session can also be easily integrated into the system.

- A “plug and play” environment for devices: Any user device cannot be used to connect to its owner unless its address is published. A user acquiring a new end-device has to publish its address to all his acquaintances. With the MobInTel architecture, the user has simply to update his/her profile with the information about the new device to be available through this device. The opposite happens when the user looses the devices or terminates the service.
- User’s privacy: a direct consequence of publishing all user’s device addresses and preferences is that the user’s privacy is jeopardized. A calling party might infer the location of his partner depending on the number he dialed. Additionally, exchanging all possible acceptable configuration parameters creates easily a privacy violation. Most users would like to keep their preferences private, and only exchange the minimum required acceptable configuration parameters. The MobInTel architecture provides location privacy to the user by using the user’s unique identifier to access all user’s devices, and the system transfers the call to the right device in a way that is transparent to the caller. To protect the privacy of user’s preferences, the MobInTel architecture extends the SIP protocol with solutions from the field of secure distributed computation. In Chapter 8, we will elaborate more on the solution and present an algorithm for privately negotiating the Quality of Service (QoS) parameters for multimedia applications without revealing the user’s preferences

7.3 Architecture of the HDA

To fully account for personal mobility, we have designed an architecture for the HDA with two major components: a Communication Agent (CA) and a QoS Selection and Negotiation Agent (QSNA) (Figure 36). We will present here a detailed description of each of these components. Additional components to support service mobility in ubiquitous computing environments are presented in Section 7.8:

– Communication Agent: The Communication Agent (CA) is responsible for the exchange of communication requests/replies with other parties’ communication agents. Communication requests/responses carry usually the content profile, use profile, and can also be used to put together the network profile as well as the profiles of all

intermediaries along the data path. Each intermediate node that forwards the request/response message adds information about its local available services as well as its network characteristics.

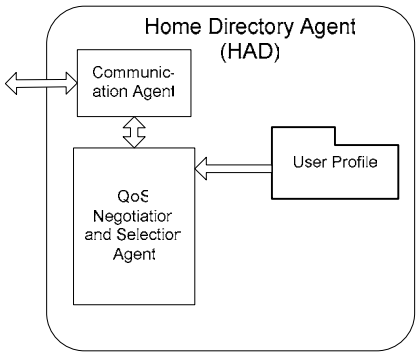

Figure 36. Components of the HDA

– QoS Selection and Negotiation Agent: The function of the QoS Selection and Negotiation Agent (QSNA) is to select the best device, QoS parameters, as well as the required adaptation services that satisfy the session requirements, and comply with the preferences of the user, and without violating the available network resources. This selection is based on the session requirements[2], the user's preferences, device capabilities, and the network and intermediaries profiles. The QSNA implements the device and QoS parameter values selection algorithm presented in the next section, which is based on the framework we have presented in Chapter 4.

7.4 Automatic device and QoS parameter selection

Before engaging in a telecommunication session, a decision has to be made as to what devices to use (camera, speaker, microphone,..), which software to start (Vic, Vat, whiteboard,..), what type of media to use (MPEG1, H.261), and the value for each QoS configuration parameter (frame rate, frame resolution, audio quality,…). This decision is affected by many factors including the hardware and software profile of the available device(s) (device profile), the available network resources (network profile), the list of available intermediate adaptation services (intermediary profile), and the preferences of the end users (user profile). This means that all these elements should be merged together

[2] Session requirements may be described using the Session Description Protocol (SDP) carried in the SIP INVITE message.

to come up with the profile for the communication session. The HDA uses the framework outlined in Chapter 4 and implements its QoS selection algorithm to decide which device to select out of all possible devices for the user, as well as the configuration parameters for the selected device and intermediate trans-coders.

As we mentioned in Chapter 4, the algorithm selects also the appropriate value for each QoS parameter for each service based on a user satisfaction function. Using all possible combinations of QoS parameters of all available services, the algorithm selects the combination of QoS parameter values that generates the maximum satisfaction within the restrictions of the devices where the services are running and the preferences of the user. The algorithm considers also all possible combinations of adaptation services available at the intermediaries along the path of the data. In addition, the algorithm always uses the network profile to make sure that the network resources required to deliver the selected combination are less than or equal to the currently available network resources.

In case that the algorithm does not find an appropriate configuration satisfying the user's preferences, the algorithm should either ask the user to release some restrictions on the preferred qualities, or it could present the user with the best configuration found, and the user is asked whether to accept the configuration or just abort the request. A similar approach was introduced in [150].

7.5 Support for session establishment in MobInTel

In this section, we will give an example to show how the HDA's in several domains cooperate between each other to establish a communication session between two users, Alice and Bob. Figure 37 shows the major steps for establishing a communication session between Bob as a call initiator, from his hotel room during a business trip, and Alice. For simplicity, we will not include the authentication part of the users. A full description of how the HDA architecture is used as a secure authentication infrastructure for mobile communication services can be found in [151,152]. We will assume that each domain has a local policy server that acts as a policy decision point for the domain. We will also assume that all users of the architecture have already registered and been assigned universal identifiers. The major steps in the session establishment are as follow:

1. Using the PC in his room, Bob submits the universal identifier and the universal identifier of Alice to the FDA.
2. To fetch Bob's profile, the FDA uses the domain name part in Bob's universal identifier and the DNS service to find the IP address of Bob's Home Directory server. The FDA also looks up the IP address of Alice's Home Directory server using Alice's universal identifier.
3. The FDA of the hotel asks the HDA of Bob for Bob's profile.
4. The FDA forwards Bob's profile, the profile of the candidate devices in the hotel that Bob can use to the HDA of Alice.
5. The HDA of Alice gathers first the network profile and the list of available intermediate trans-coding services from the network. Next, it combines Bob's profile and his candidate devices, Alice's profile and her candidate devices, together with the network and intermediaries profiles as we described in Chapter 4 to determine the devices that best suit the preferences of Alice and Bob, as well as the multimedia QoS parameters to be used by these devices. The system also will generate a list of requirements to be carried out before the session can be established. An example of a requirement would be to use zero or more trans-coders between the two end points. Trans-coders, if required, are instantiated and the session is established between the selected devices of Alice and Bob (Figure 37).

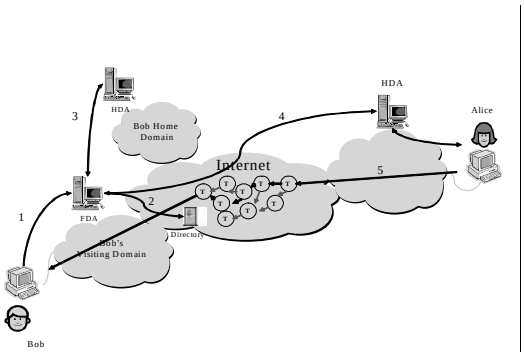

Figure 37: HDA support for session establishment.

6. **Figure 38** is an example where the HDA decides that the best way to connect Alice and Bob would be through the phone. The HDA locates a

PINT server [153] and passes the phone numbers of Alice and Bob to the PINT server to connect them over the Public Switched Telephony Network (PSTN).

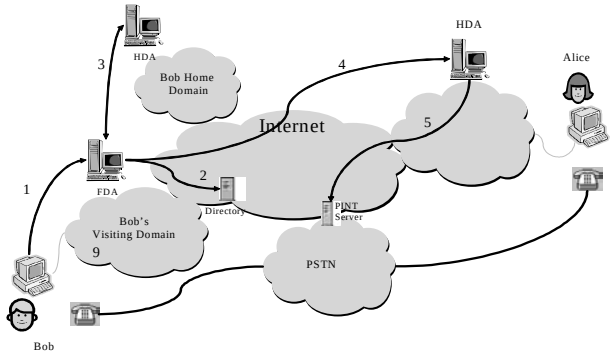

Figure 38. HDA support for connecting two PSTN phones.

7.6 Extending MobInTel to support personal and service mobility in ubiquitous computing environments

Ubiquitous computing is a new trend in computation and communication; it is at the intersection of several technologies, including embedded systems, service discovery, wireless networking and personal computing technologies. It is best described by Mark Weiser, father of ubiquitous computing, as the world with “invisible” machines; a world such that “its highest ideal is to make a computer so imbedded, so fitting, so natural, that we use it without even thinking about it”[154]. In such an environment, computing devices are shifted to the background, and they are only visible through the services they provide; specific information about these devices such as location, address, or configuration parameters are totally transparent to the user.

One of the major contributing factors to the big interest in ubiquitous computing is the advance in short-range radio frequency communication. This advance has created the notion of personal-level communication infrastructure, referred to as Wireless Personal Area Networking (WPAN), of which Bluetooth [155] is an example. Devices connected over WPAN have the capability to locate, communicate, and provide services for each other. This capability allows these devices to collaboratively provide an ad hoc distributed computing environment and to deliver services that cannot be possibly

delivered with only one device. For instance, a video display service may look for an audio playing service in its vicinity to play an audio and video recording. Also, an audio play-out service can use the service of a computer connected to the Internet to download music from the Internet before playing it out.

Given these trends in personal communication, there is a growing need to provide personal and service mobility for persons roaming in ubiquitous computing environments. Personal mobility [156] is defined as the ability of a user to get access to telecommunication services from any terminal (e.g. workstations, notebooks, Personal Digital Assistants (PDA), cellular phones) at any time and from any place based on a unique identifier of the user, and the capability of the network to provide services in accordance with the user's service profile. Closely related to the subject of personal mobility is service or session mobility [138], which refers to the possibility of suspending a running service on a device and picking it up on another device at the same point where it was suspended. An example of service mobility is a call transfer from the mobile phone of the user to his office phone.

Throughout the rest chapter, we will show how an extended MobInTel architecture could be used during a communication session between Alice, a team manager on a business trip, and her team-members. The elaboration of the scenario is presented in Section 7.10. Before the meeting, Alice would like to have a small chat with Bob, who is a team leader in her group. Using the multimedia workstation in the business office of the hotel, Alice sends an invitation for a multimedia conversation to Bob, ten minutes before the meeting starts. Bob, sitting in his office, receives the invitation on his PDA. Since Bob has indicated in his profile that he is always willing to accept calls from Alice, his PDA tries to find a microphone, a speaker, a video display service and a camera to make for a full multimedia session. Assuming that such services exist in Bob’s surrounding, the PDA discovers and reserves these services for the communication session with Alice. The PDA sends back information about all these services, and the videoconference is started between Alice and Bob.

When it is time for the meeting, Bob moves with his PDA into the conference room where all the team members are waiting. Bob’s PDA detects that the services that Bob was using are not available anymore, and since he has already set the FOLLOW-ME

option of the session to ON, his PDA tries to discover similar services to continue the session in the conference room. The PDA then detects and selects the big screen, the camera, the speaker as well as the microphone of the conference room; Alice's picture appears on the big screen and she is now ready to participate in the meeting.

7.7 Architectures for personal and service mobility in ubiquitous computing environments

In Chapter 6, we have reviewed a number of architectures [9,49,135,136,137,138,157] that have been proposed to solve the problem of personal mobility in the context of the Internet. All these architectures share the same concept of a Directory Service (DS) that provides a user-customized mapping from a unique user identifier to the device that is best suitable for the user to use. The basic idea of these architectures is that the user keeps a profile in the DS containing static information about all the devices he/she has access to, and the logic or policy (user preferences) for when to use these devices. When the DS receives a session initiation request for a user, it executes the logic in the user's profile and handles the request according to the user's specified preferences.

While these architectures provide personal mobility for users with multiple telecommunication devices (telephone, PDA, laptop, cellular phone, pager), they all fail short to extend personal mobility to ubiquitous computing environments because:

- the information in the Directory Service is static,
- there is no support for service discovery,
- they lack support for service mobility, and finally
- they lack support for complex (combined) services.

A number of research works have addressed the problem of service discovery and selection in ubiquitous computing environments. The work in [158] presented two approaches for selecting services based on the physical proximity and line-of-sight of the handheld device relative to the service. The authors in [159] used a central gateway that makes the decision of delegating rendering tasks to devices in the environment of the user. A PDA carried by the user is responsible for detecting the user's nearby devices, and sending the list of available devices to the gateway. Both these two architectures

[158,159] suffer from the drawback of using infrared communication for finding and/or selecting services. Because infrared communication requires aligning the devices before any communication is established, these architectures cannot be used in ubiquitous environments because they require user's awareness of the location of devices. Moreover, service mobility and QoS issues were not discussed in these works. Beigl *et. al.* [160] investigated the use of a browser running on a PDA to enable ubiquitous access to local resources as well as resources on the World Wide Web. The browser, called the Ubicompbrowser, detects devices and resources in the environment of the user, and delegates the rendering of the requested resources to the nearby devices in order to overcome the limitation of the PDA. A major drawback of the Ubicompbrowser is that it requires the user to know its current location to select the rendering devices. Additionally, the Ubicompbrowser does not deal with the issue of QoS negotiation, neither with the issue of service mobility. Campadello [157] addressed the problem of change in the execution environment of applications through the composition of basic hardware and software building components available in the execution environment. Based on the preferences of the user and the application logic, a personal agent dynamically combines hardware and software building components available in the current execution environment to build an instance of the requested application. Campadello's architecture suffers from the same drawbacks as the Ubicompbrowser and does not also consider the possible adaptation space at the content level.

In a recent project, researchers at the Smart Space Laboratory (SSLab) [161] have suggested the use of embedded computers as a substitute for the awkward interface of the portable devices. The researchers have suggested that collaboration between a portable device and embedded computers can help alleviate the problem of limited input and output capability on the portable device [162]. They have demonstrated their approach by implementing a mobile TV-phone prototype that uses a nearby high resolution-display instead of the small display on the mobile phone.

Our extension to the MobInTel architecture is different from all these reviewed architectures in that it dynamically discovers and updates the list of services available in the WPAN of the user. Our architecture uses also a QoS negotiation and selection algorithm to select the services that best suit the context and preferences of the user. The

architecture can also mix-and-match different services to fulfill all the requirements of the session. Additionally, our architecture incorporates additional components in order to support smooth and transparent service mobility.

7.8 Extension to the MobInTel architecture

The extension of the MobInTel architecture was inspired by the Ubicompbrowser project, and is intended to support personal mobility in ubiquitous environments. The extended architecture includes additional functionalities to overcome its shortcomings in ubiquitous environments. The modified architecture uses the short-range Bluetooth wireless communication to construct the user's WPAN, and to restrict the domain of services available to the user just to the services running on devices that are within this WPAN. Our architecture differs also from the architecture in [158, 159] in that service selection is done automatically on behalf and according to the preferences of the user, and without requiring the user to point and select each service individually using infrared, (since the user might not, and should not, be aware of the services and their locations). We also address the problem of service mobility by using periodical search for services similar[3] to the services currently used by the user, in order to provide smooth hand-off for these services.

In a typical ubiquitous computing environment, the set of available devices for the user may change continuously as the user changes his/her location. Updating manually the information about currently available services is not an option. Additionally, discovering the available services and sending update messages to the HDA is not a practical solution either, since the set of available services might change very often with the environment, which results in many update messages sent to the HDA. Moreover, if the update message incurs a certain delay, the information included in the message could be outdated when it gets to the HDA.

To overcome these limitations, we propose to run a modified version of the HDA on a hand-held device, such as a PDA, that is always carried by the user. We call this modified version of the HDA the Personal Agent (PA) of the user, and it is responsible for detecting devices in the vicinity of the user as well as managing the user's

[3] We say that two services are similar if they serve the same purpose, for instance a TV and a wall projector, or a PC speakers and a mini-stereo.

communication sessions. In order to retrieve the user profile and send/receive communication requests through the HDA, we require that the hand-held device, on which the Personal Agent runs, to have access to the Internet (through a wireless modem or IEEE 802.11[163] connection). The PDA is also supposed to be able to join a Wireless Personal Area Network (such as Bluetooth WPAN) in order to be able to detect and communicate with other wireless devices just around the user. These requirements are readily available, for instance, in the new iPAQ Pocket PC models from Compaq. For the rest of the chapter, we will assume that the PA is running on a PDA that satisfies these communication requirements.

At any one time, either the HDA or the PA is responsible for providing personal mobility service to the user. When the PDA is switched ON, the PA contacts the HDA to retrieve the user profile. From that point on until the PDA is switched OFF, the PA is responsible for executing the logic in the user profile, and the HDA would switch into passive mode and act only as a proxy for incoming call requests. To ensure that the HDA is aware of the status of the PA, we decided to send all replies to communication requests through the HDA. The HDA can detect when the PA is not running or the PDA is currently out of reach if the HDA does not see a reply to a forwarded call after a certain time-out period. The HDA would then switch into active mode, and handle the communication request according to the rules specified in its local copy of the user profile.

To fully account for personal and service mobility in ubiquitous computing environment, The PA includes all the components of the HDA with three additional components; these additional components are: a Service Discovery Agent (SDA), a User Context Agent (UCA), and a Service Registry (SR). Figure 39 shows the architecture of the Personal Agent with its components. We will present here a detailed description of each of these components.

– Service Discovery Agent and Service Registry: The function of the Service Discovery Agent (SDA) is to search for all services in the WPAN of the user. Because devices in ubiquitous computing environment are more likely to be single-service devices, the line between a device and a service becomes blurred. For instance, a display device can be identified by a display service with the same capabilities as the physical device that

could be discovered using service discovery protocols. We will refer hereafter to the device that provides a visible service to the user as an end-service, and the rest of the services as intermediary services. Example of an end-service is a video display screen or an audio play-out service.

The SDA provides the QoS Selection and Negotiation Agent (QSNA) (discussed below) with the list of currently available services (including end-services). Since different services might be using different service discovery protocols (JINI [102], SDP [164], or SLP [103]), the SDA shall act as a service discovery client in multiple service discovery protocols. The SDA periodically searches for all available services, and stores this information in the Service Registry (SR). This information allows for fast session setup and smooth service mobility, as we will discuss in Section 7.9.

– User Context Agent: The User Context Agent (UCA) is responsible for collecting and providing the user's context profile to the QSNA. Context information includes, as we mentioned earlier, the location of the user, whether the user is by herself or surrounded by other people [160,165], and any other context information. The UCA assists the QSNA during the service selection phase by providing up-to-the-minute information about the user's context.

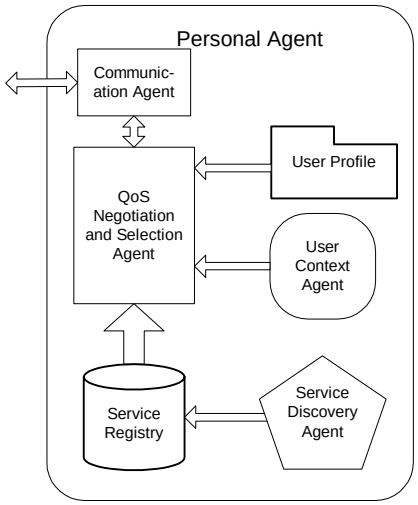

Figure 39. Components of the Personal Agent

As with the MobInTel architecture, the QSNA makes the decision on which service out of all the available services in the vicinity of the user to use, based only on the

preferences from the user profile, the user's current context, and within the availability and capability of the end-services in the vicinity of the user.

7.9 Support for service mobility

During a communication session, a nomadic user in a ubiquitous environment might move away from one device and get closer to another device that provides a service similar to the one used on the first device. To continue with the communication session, the user has to re-initiate the session again with the new services. This could be a problem for the user, especially if the user continues moving from one place to the other during the session. Our architecture solves this problem by supporting service mobility during the communication session.

Service mobility is required since the life span of the communication session might be longer than the time during which the currently used device is available in the user's WPAN. To solve this problem, the Personal Agent should switch the service from one device to another device providing a similar service when the device that is currently used becomes unavailable or it should inform the user about the disappearance of the service. For instance, a user moving away from his computer and entering the conference room should have, if desired, the multimedia session transferred from his computer to the TV and stereo system in the conference room. If the conference room does not have a TV set, the user should be warned that the video display service would be discontinued if he/she stays in the conference room.

Our architecture supports service mobility through service hand-off, transparently to the user. A smooth transparent service handoff requires continuous discovery and update of the Service Registry (SR) with information of the currently available services in order to provide smooth service transfer. When a connection to a service is fading, the SDA informs the QSNA about the possible replacement service. The QSNA passes the information about the new service to the Communication Agent, which in turn, sends an update message the Communication Agent of the other party.

7.10 Usage scenario (continued)

In this section, we will elaborate more on the scenario presented in Section 7.7. We will assume that the SIP [49] signaling protocol is used to establish and maintain the communication session. We also assume that a SIP *invite* message can carry the call initiator profile as well as the profile of his/her devices as a payload. The scenario of Alice trying to reach Bob, who is in the lounge area, is divided into five phases (Figure 40), with the first phase executed only once, when Bob switches **ON** his PDA. We will assume that Bob has enabled the service mobility option with his Personal Agent. Due to the space limitation, we will only give a short description of each phase:

- **Startup Phase:** The Personal Agent retrieves the user's profile from the Home Directory Agent **(Messages 1-2)**.
- **Session Initiation Phase:** Alice's agent (a HDA agent or PA agent) sends a SIP *invite* message to Bob's Home Directory Agent containing the profile of Alice as well as her device profile. Bob's Home Directory Agent (HDA) forwards the request to the Personal Agent **(Messages 3-5)**. The SDA uses the service discovery protocol to discover the available services for the session and update the SR (**Messages 6-8).** The QSNA selects from the SR the services for the session based on the session requirements, the user profile, network and intermediaries, and device/service profile, as described in Chapter 4. The QSNA might also mix-and-match several devices to provide compound services. Bob's Personal Agent sends back to Alice's Personal Agent the information about the selected services. **(Messages 9-11)**
- **Data Exchange Phase:** The data is exchanged between Alice's device and the selected devices from Bob's environment.
- **Session Maintenance Phase**: As long as the session is still running, the SDA periodically queries the environment for services that are similar to the services used in the session. This information is used to update the SR in order to reduce the delay in service mobility (as we discussed in Section 7.9). When Bob moves to the conference room, the SDA detects the audio and video services of the conference room. **(Messages 12-14)**
- **Service Hand-off Phase:** In case a service that is currently used becomes unavailable because of the mobility of the user, the SDA informs the QSNA of the replacement

service(s) (in this scenario, the replacement services are the services of the conference room). The QSNA in turns sends an update message through the CA to Alice's PA with the information of the new services. **(Messages 15-16)**

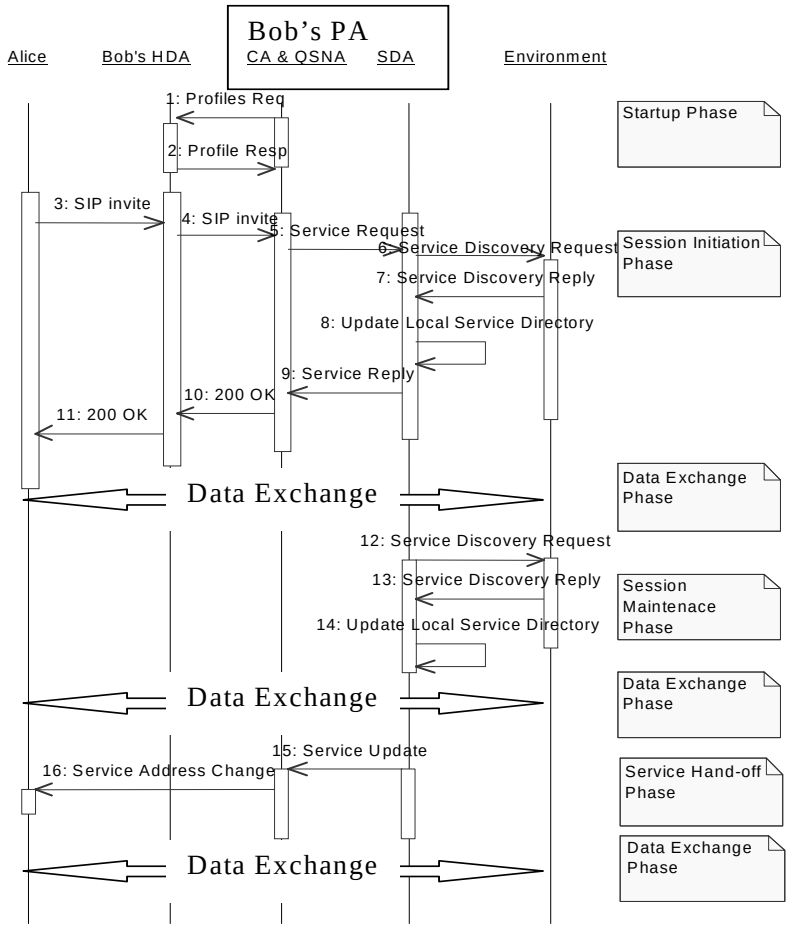

Figure 40. Session establishment based on the Personal Agent

7.11 Experimentation and evaluation

In order to get a better understanding of the system, we have developed a prototype of the extended MobInTel architecture in our lab [166]. We wanted to show how to provide personal and service mobility to the user, and in case of multiple similar services, how the selection algorithm can correctly select the service that is more satisfactory to the user. The prototype allowed us also to measure the performance of the system in order to study the feasibility of the system.

7.11.1 Hardware, software, and communication protocols

In this section, we will present the hardware and software platform for the prototype, the communication protocols, as well as the architecture software components of the prototype.

- **Hardware**: For the hardware platform of the prototype, we have used a number of devices, including:
  - **a.** Two IBM Pentium III 1600 MHz PC's with 128 MB RAM and running Windows 2000 operating systems. Each PC is hosting one audio and one video service. Each of the PC's is equipped also with a Bluetooth device connected on the USB port. Both PC's run Linux with BlueZ module for the Bluetooth hardware (3COM device) and connected to the Internet.
  - **b.** An IBM ThinkPad T40 laptop to host the caller side and the media delivery server. The laptop has a video camera and a microphone connected to it and functions as a multi-media streaming server, capturing the audio and video data and delivering it to the selected services running on the PC's.
  - **c.** A Compaq's handheld iPAQ 3870 PocketPC device to run the Personal Agent. The iPAQ 3870 runs Windows CE by default; but, in order to use the BlueZ Bluetooth stack, we installed Linux on the iPAQ. The iPAQ has also a wireless 802.11 card that connects the iPAQ through an 802.11 hub to the Internet. All control messages to the iPAQ are exchange through the wireless hub.
  - **d.** For the transcoding infrastructure, we have used three PC's (as intermediaries) running the transcoding services on top of JMF.

- **Software and services:** We have also used a number of software to integrate and test the prototype of this thesis:
  - J2SE (*Java 2 Standard Edition*) version 1.3.1 to develop caller side software on the laptop and the HDA's.
  - Jeode: Java Virtual Machine that runs on the PDA from Insignia's to develop the PA software for the PDA.

- JMF (*Java Media Framework*) version 2.1.1a. for the transcoding and streaming of data. Table 9 shows the list of all the JMF trans-coders used and their supported formats.
- SDP server: To enable service discovery, each machine runs a Bluetooth Service Discovery Protocol (SDP) server to manage local services on the machine. The SDP server acts as a directory lookup server: it keeps records of registered services, receives requests and sends replies about the registered services.

Table 9. Transcoders used in the prototype

Transcoder	Hosting Proxy	Supported Source Format	Supported Destination Format
Transcoder 11	Proxy 1	MJPEG 640x480	MJPEG 320x240
Transcoder 12	Proxy 1	MJPEG 640x480	h263 704x576
Transcoder 13	Proxy 1	h263 176x144	h263 128x96
Transcoder 21	Proxy 2	MJPEG 320x240	MJPEG 160x120
Transcoder 22	Proxy 2	h263 176x144	h263 352x288
Transcoder 23	Proxy 2	MJPEG 320x240	MJPEG 640x480
Transcoder 31	Proxy 3	h263 352x288	h263 176x144
Transcoder 32	Proxy 3	h263 704x576	MJPEG 640x480
Transcoder 33	Proxy 3	MJPEG 160x120	MJPEG 80x60
Transcoder 34	Proxy 3	h263 128x96	MJPEG 80x60

– **Communication protocol:** As for the communication protocol, we have used the following:
  - Session Initiation Protocol (SIP): For setting up, modifying, and tearing down the communication session, we have developed a small version of the SIP protocol (6.3.5) with only minor communication messages.
  - SDPTool: We have used the program SDPtool [167] on the PDA to check which services are made available by the nearby devices.

7.11.2 Experimental environment

Our experimental environment consisted of a large room, with four services: two audio play-out services and two video display services as shown in Figure 41; Audio Service 2 (AS2) has a better audio quality than Audio Service 1(AS1), and Video Service 2 (VS2) has a better quality (higher resolution) than Video Service 1 (VS1). We have

selected two locations in the room, location 1 and location 2, and the user of the system can move in-between the two locations. While standing in location 1, the user's PDA can discover only Audio Service 1 and Video Service 1, and while the user is in location 2, the PDA can discover all four services.

We start the experiment with the user in location 1 when he receives an incoming call through the 802.11b wireless interface of the PDA. The call requires an audio service and a video display service. The PA on the PDA analyses the request and, locates the two services, Audio Service 1 and Video Service 1. The Communication Agent sends a message to the caller's Communication Agent (CA) containing the information of these services. The caller's media server sends the data directly to these services.

When the user moves to location 2, two better services, Audio Service 2 and Video Service 2 also become available to the user. The PA discovers the new services, and based on the user's profile, decides to use the new services since they are more satisfactory to the user. The Communication Agent sends an update message to the caller's PA with the information of the new services, and the audio and video are now switched to Audio Service 2 and Video Service 2. When the user moves back to location 1, Audio Service 2 and Video Service 2 become unavailable again, and the data is sent again to Audio Service 1 and Video Service 1.

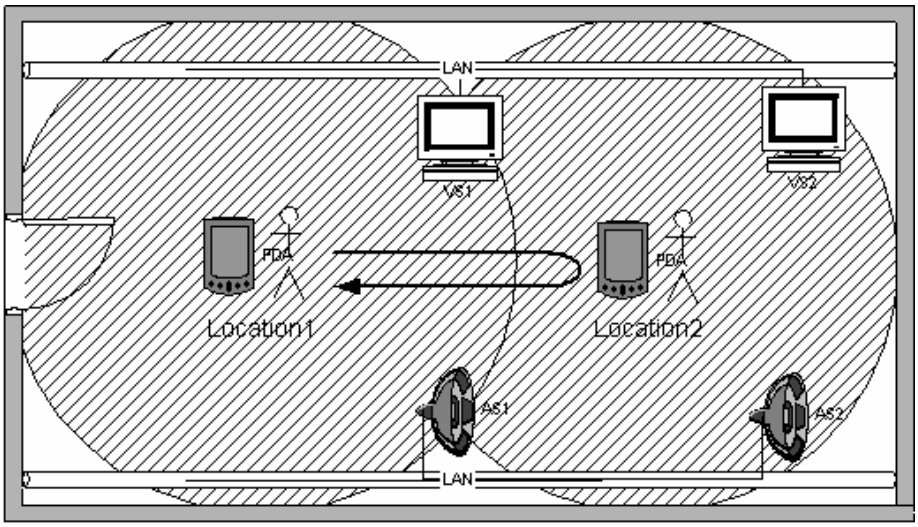

Figure 41. Experiment environment layout

7.11.3 Results

After building the prototype, we were able to collect a number of performance metrics. We placed the media server of the caller and the audio and video services on the same local area network (LAN), while the PDA was connected to the LAN through an 802.11b wireless access point. We were interested in measuring the following metrics:

Total Signaling Time: The time it takes for the signaling messages to travel from the caller to the callee and return, including the time to select the QoS parameters and intermediate adaptation services.

Media Server Initialization Time: The time it takes the Communication Agent (CA) of the caller to read the reply message from the network and to initialize the Java Media Framework audio and video servers, responsible for capturing and sending the audio and video data.

Data Transfer and End-Service Initialization Time: The total time required to initiate the sending and receiving services and the data transfer delay between the two services. These parameters where combined together since this is the time observed by the user of the system.

We have performed five run tests, and the average results (in seconds) are presented in Table 10.

Table 10. Session setup time

	Total Signaling Time	**Media Server Initialization Time**	**Data Transfer & End-Service Initialization Time**	**Total Setup Time**
Time (sec)	2.2	5.3	2	9.5

The Data Transfer and End-Service Initialization Time included in the table does not include any delay incurred by intermediate adaptation services. We also note that some additional delay (not mentioned in the above table) can be expected due to service mobility. This delay is a result of the fact that the SDA runs periodically, and not continuously. In our prototype, we set the discovery agent to run periodically every 30 seconds, to balance between the incurred computation load on the PDA and the speed and accuracy of service discovery. Additionally, the SDA spends 10 seconds, using the SDPtool, trying to discover all the available services. In this setting, the SDPtool

consumes around 18% of the CPU resources when running on the iPAQ 3870 Compaq handheld with 206 MHz ARM processor. As a result, the total service mobility time could vary between 19.5 sec (10 seconds for the service discovery and 9.5 for the signaling and data transfer) if the user reached the new location right before the service discovery agent started the periodic check, compared to 49.5 seconds if the user moved into the new location right after the discovery agent had finished the service discovery process.

We should mention that for an IP telephony service to be accepted by users, the time delay of the service should be equal if not better than the plain old telephony service. We acknowledge that the measured delays in our prototype are large, but there are a number of ways to improve the performance in a streamlined implementation. The first improvement would be to replace the freeware SDPtool with another faster implementation of the native SDP protocol. Even the native SDP tool suffers from long service discovery delays since each node is obliged to establish a connection with every other node before it can perform service discovery. Some existing work [168] promises to reduce the service discovery time in Bluetooth.

Additionally, we can also improve the Data Transfer and End-Service Initialization Time. This delay results mainly from the time it takes to initialize and start the end-service. In our prototype, we just register the end-services, but never initialize them. This initialization time could be virtually eliminated if the services are already up and running on the device.

Finally, our current implementation does not take the capabilities (battery, CPU, and memory capacity) of the PDA into consideration when executing the code for the PA. Depletion in any of these resources would definitely affect the performance of the service and personal mobility. One could foresee that the service discovery period becomes a function of the available resources. Another approach would be to let the user switch off the service discovery when he/she is not moving. The PA may also run the service discovery agent less frequently when there has not been a recent change in the discovered services.

7.12 Conclusion

In this chapter, we have presented our MobInTel architecture for personal mobility. The architecture uses the QoS selection algorithm presented in Chapter 4 to select the communication quality for mobile users based on device capabilities and user preferences. Using this scheme, user mobility becomes completely transparent for the mobile user. We have also presented an extension to the architecture to support personal mobility in ubiquitous environments. The extended architecture allows nomadic users to benefit from the availability of large number of hidden services in a ubiquitous environment to establish communication sessions. To construct this architecture, we introduced a new component that we called the Personal Agent (PA) that acts on behalf of the user during the service discovery and selection process. The Personal Agent also provides support for service mobility through periodic updates of currently available services into a local service registry. We have also shown the functionality of the Personal Agent during a typical communication session using an example scenario. Additionally, we have presented a prototype of the architecture and some performance measurements. We have also discussed a number of recommendations to improve the performance of the prototype. In future work, we are planning to implement the recommendations before we can study the usability of the architecture.

Chapter 8

Preserving the Privacy of User Preferences in Multimedia Communication

8.1 Introduction

Over the past few years, there has been a widespread increase in the use of Internet as a platform for wide range of applications in different areas including medicine, education, commerce and telecommunications. Most of the time, these applications require some personal or financial information from their consumers. Such information could be used for a number of purposes, ranging from regulating access to on-line services (authentication, authorization), to billing (accounting), to service maintenance, customization or adaptation.

For Internet users, the temptations to disclose personal information are numerous, including the convenience of putting orders online, the privileged access to some resources, and the benefits of personalized and value-added services. But incidents with online privacy violations [169] made most online users concerned with the privacy invasion risk associated with revealing personal information. The risk has lead Internet users to conceal true information or to "garbage in" fake data in order to bypass the information request phase. A number of identity management systems [170,171,172] and software proxies [173,174] already exist. These systems help their users to control how much personal identifiable information is released. Concealing true information by using identity management and proxy systems or even providing fake information is acceptable for web surfing, but not for many other applications such as IP-telephony or video-conference, where genuine user's identity and preferences are crucial inputs to the application.

IP-telephony and videoconference applications are more of a personal communication type of applications where users have to identify themselves to each other. More importantly, users of these applications have the choice to fine-tune a number of configuration parameters in the application to suit best their preferences. For

instance, an employee can specify, that when using a video-conference application to talk to a senior manager or client, to use CD-quality audio media and 30 frames/second video media with large screen resolution, while limiting the audio quality to telephony-quality with no video media when talking to a co-worker. These configuration parameters, or as we shall call them user's preferences, are usually exchanged between the two communicating parties before establishing a communication session.

Exchanging all possible acceptable configuration parameters creates easily a privacy violation. Most users would argue that these preferences are private information and shall be protected. Additionally, during session establishment, users would like to keep their preferences private, and only exchange the minimum required acceptable configuration parameter values. While many security schemes can be used to protect against unauthorized access to the user's preferences, a solution is required to protect the privacy of user's preference during session establishment, which is the subject of this chapter.

The remainder of the chapter is organized as follows: in Section 8.2, we will present a literature review on the issue of privacy protection over the Internet, followed in Section 8.3, by a closer analysis of privacy invasion when using multimedia applications. We then outline three solutions to the problem of privacy invasion, in Section 8.4: the first solution is based on the use of trusted third party, the second solution is based on the notion of trust between communicating parties, and the third solution is based on protocols from the field of secure distributed computation. In Section 8.5, we elaborate more on the third solution and present an algorithm for privately negotiating the Quality of Service (QoS) parameters for multimedia applications without revealing the user preferences. An analysis of the computation and communication cost of the algorithm is presented in Section 8.6, and finally our conclusion and future work are outlined in Section 8.7.

8.2 Literature review

The issue of protecting user's privacy over the Internet has been a hot research topic over the past few years. Of major interest is the work by the World Wide Web Consortium (W3C) on the Platform for Privacy Preferences (P3P) [175]. P3P is a

specification language designed to inform users about the privacy policies of visited web sites. When a P3P compliant client browser requests a resource, the web service replies with a machine-readable privacy policy, which includes a declaration of the service identity and its privacy practice. The privacy practice lists the data elements that the service proposes to collect, how each data will be used, how long the data will be retained for and with whom it is shared. Acting on behalf of the user, a user agent can parse the declared privacy policy and compares it against a set of privacy preferences defined by the user. The user's preferences are expressed as a set of rules using the APPEL specification language [176]. The result of the comparison might be either to proceed unconditionally with the request, or to proceed with the request and to provide the minimum information required to still get the requested resource, or to block completely the request. Tamaru *et. al.* [177] used the privacy policy of online services provided in public places to evaluate user's trust in these services. Based on the evaluated trust level, a user can change the granularity of his/her profile before sharing it with the service. The Customer Profile Exchange (CPExchange) also uses the P3P specification to express user's privacy preferences

Rezgui *et. al.* [178] presented an approach for preserving privacy in government web services. The approach is based on the use of digital credentials, data filters, and mobile privacy enforcement agents. The architecture requires users to have the right credential to access web services. Data filters use also these credentials to protect the security and privacy of data requested from the user, in conformance with the preferences of the data owner. In order to protect the privacy of the data when exchanged with a third party, the architecture encapsulates the data in a privacy enabling agent that enforces the privacy rules of the data while on remote systems.

Zhang and Wong [179] presented an agent-based negotiation architecture that uses Case-Based Reasoning technique to capture and re-use previously successful negotiation experiences in the course of current negotiation session. Negotiation agents can use the information about previous negotiation to decide on the negotiation strategy for each episode of the negotiation. Negotiation experiences are hierarchically arranged and similarity between experiences is based on the concessions made during the negotiation. An approach for bilateral negotiation between an e-service provider and an

e-service consumer in the presence of uncertainty is presented in [180]. During the negotiation, an agent makes use of the experience of other reputable agents to make an offer or a counteroffer to each other. Seamons *et. al.* [181] presented an approach for automated trust establishment, where trust between interacting parties is incrementally established by exchanging credentials and requests for credentials. This interactive approach, known as trust negotiation, controls the exact amount of information exchanged during trust establishment. But the approach does not support interaction to determine the best user's preference.

While these previous architectures can solve the problem of protecting online users' privacy when dealing with web services, they do not provide solutions for personal multimedia applications; these applications require the user's identity and preferences to establish multimedia communication sessions. In the next section, we will present three solutions to the problem of negotiating the configuration parameters for multimedia communication sessions, like video-conference, while at the same time preserve the privacy of the participants' preferences.

8.3 Privacy invasion in multimedia applications

Multimedia applications usually allow the users to fine-tune a number of configuration parameters in order to get the best reasonable result. In Chapter 6, we have presented two architectures to support customized multimedia communication applications, where each user has a profile to store his/her preferences for a number of configuration parameters.

Generally speaking, a user's profile would capture the personal properties and preferences of the user. Properties include, but are not restricted to full name, employer, contacts (email account, phone number…), list of devices and services subscribed to by the user, etc... As for preferences, they usually cover the application level user's preference for all applications to which the user has access to, particularly the user's preferences concerning the quality of multimedia communication. These preferences would usually cover the choice and preference of receiving and sending audio and video media such as the frame rate, the resolution, the color depth the audio quality and the degree of preference (weight) for every media types.

User's preferences expressed in the profile can be classified as application layer QoS parameters. To better describe these parameters from the user's high-level perspective, a number of frameworks [182,183,184,185,186,187,188] already exist that map these application layer QoS parameters into the expected end user experience, or what is defined as Quality of Experience (QoE). Siller and Woods [185] defined the QoE as the "user's perceived experience of what is presented by the application layer".

In Chapter 4, we have used a quantitative metric of the QoE, called the user's satisfaction[4]. While this definition is suitable for defining the user satisfaction or experience, it applies only to data consumers and ignores completely data producers. This model is suitable for the type of applications where the session parameters are fixed by the session initiator, such as loosely coupled conferences over the Mbone, and not negotiated by the session participants, which is not the case with interactive personal applications. For these applications, participants are much more directly involved in the communication and any session management protocol for these interactive applications strives to maximize the QoE of the data consumers but at the same time, keep the control of what is delivered to the consumer in the hand of the data producer. We term this fact as the receiver's QoE and the sender's control.

In order to accommodate for the concept of receiver's QoE and the sender's control, we require that each user should have two different satisfaction functions: one for expressing his/her preferences with the data producing or sending values, and the other to express his/her preferences with the data consumer receiving values. During the session setup phase, and with each direction of media flow, the session management protocol would try to maximize the receiver's satisfaction value with each configuration parameter, without exceeding the senders' allocated bandwidth and without causing his/her satisfaction value to go down to zero.

Selecting the QoS parameter value that maximize the receiver's QoE can be carried out at the sender's side if he/she has a complete knowledge of the receiver's preferences and his/her satisfactions functions(s). But simply passing the receiver's information to the sender presents privacy risk to the receiver who might consider these

[4] For the rest of the chapter, we will use the term QoE to represent the user's satisfaction or user's experience.

parameters as private information. For instance, if Bob wants to communicate with Alice, Bob would most probably prefer not to share with Alice his preference of 10-20 frames per second, and would prefer to pass to Alice only one value to use for the session, which he is sure that Alice does not mind providing. Also, Bob would definitely prefer that he decides on the value to use, rather than delegating this personal task to Alice. Additionally, when Alice selects the value to use, there are no guarantees for Bob that Alice really selected the best value for him. Giving Alice all Bob's acceptable preferences and asking Alice to select the value to use is similar to showing all credential cards to somebody who wants to verify your identity and asking him to select the card the verifier want [181]. This is where the need for a protocol that controls the privacy of the interactive multimedia applications is required. Our work here proposes three solutions to address the problem, all of them presented in the next section.

8.4 Proposed solutions

In this section, we will present three solutions for the problem of privacy breach when negotiating the configuration parameters for multimedia applications. The first solution assumes the existence of a third trusted party; this solution is the easiest and most trivial one, but highly unlikely adopted approach. The second solution is based on the concept of "notify and consent", that does not require a third trusted party, but instead requires a certain degree of mutual trust between the communicating parties. The third solution is the strictest one, and is based on some primitives from the field of secure distributed computation.

8.4.1 Using a trusted third party

Protecting user's preferences during communication session was never an issue when using the Public Switched Telephony Network (PSTN), simply because of the communicating model used. The PSTN communication style is based on a 3-tier model, where the users connect to each other using trusted third party, which is the PSTN service provider; signaling messages are exchanged only between the end-users and the services providers with no end-to-end signaling. Internet communication is more of a peer-to-peer model where communicating parties communicate directly with each other, without a

third party in the middle: the signaling messages are generated and consumed by the end-parties, and not by any intermediate service provider.

The problem of computing common preferences for a multimedia session can be trivial if there could be a jointly trusted third party, to whom all communicating parties would pass their profiles. This jointly trusted third party would compute the common acceptable session parameter without leaking any information. Such approach is easily implemented with the H.323 protocol [48], which provides an optional Multipoint Control Unit (MCU) for multiparty conference control. However, the Internet environment is more of peer-to-peer environment, and for any solution to be widely adopted, it must fit this peer-to-peer model. Additionally, it can be difficult for all communicating parties in a session to agree on a common trusted third party.

8.4.2 Using P3P/APPEL to announce privacy policy and build trust

A second approach to keep user's privacy could be based on the concept of mutual trust among peer parties. In this approach, each party can announce what information from the user's profile is collecting and for which purpose. A second party can, before revealing the information in the profile, download the privacy policy and compare it to locally expressed privacy rules. If the conditions and terms of the privacy policy do not violate the user's privacy rules, only then the requested information in the profile could be passed to the other party. The P3P [175] specification language and its APPEL [176] companion language can be used to express respectively the user's privacy rules and preferences.

This approach has already been suggested in [189], where personal identifiable information included in the Composite Capability/Preference Profile (CC/PP) [190] has been protected by using CC/PP with P3P (Platform for Privacy Preferences). Similarly, Tamaru *et. al*. [177] showed how to use information in the P3P policy to assign a trust-level to a website. Using the computed trust-level, a filtering process is used to enforce the user's privacy. Zandbelt *et. al.* [191] presented a mechanism that uses a proxy server to manage the security and privacy of user's personal information. Using certificates and access control lists, the proxy allows a profile requester to see only what the user allows them to see.

8.4.3 Using distributed secure computation

The ubiquity of the Internet has opened a wide opportunities for distributed cooperative computations, where multiple people work together to solve a single problem. Due to the nature of distribution and collaboration, it might be possible that these parties do not trust each other, but they still want to perform a joint task. This set of problems is commonly referred to as distributed secure computation.

In the next section, we will show how to use some well-accepted primitives in distributed secure computation, such as secure 1-out-of-2 Oblivious Transfer (OT_1^2) and Oblivious Polynomial Evaluation (OPE), to construct a protocol for privately negotiating the QoS parameters for multimedia applications. The protocol will ensure that the two communicating parties can negotiate the QoS to use for the application without revealing all the user's preferences for these parameters. Only the resulting parameters agreed for the session are revealed without jeopardizing the privacy of all the user's preferences.

8.5 Using oblivious polynomial evaluation to negotiate private preferences

Oblivious Polynomial Evaluation is a security protocol that allows two parties, Alice with a polynomial P(x) and Bob with a variable x_*, to compute $P(x_*)$ without Alice learning the value of x_* and without Bob learning $P(x)$. In this section, we will show how we can use Oblivious Polynomial Evaluation (OPE) to find the QoS parameter value that satisfies the user most, without violating the sender's preferences or exceeding his/her allocated bandwidth. Each receiver can use Oblivious Polynomial Evaluation (OPE) to verify that the sender's satisfaction with the selected QoS parameter value is different from zero. But before we can talk about OPE and how we will use it to privately negotiate the QoS parameters, we will introduce first the notion of Oblivious Transfer (OT), which is a major building block for the version of OPE that we use.

8.5.1 Oblivious Transfer

Oblivious Transfer (OT) is a cryptographic primitive that was first introduced by Halpern *et. al.* in [192]. With different existing variants, 1-out-of-2 Oblivious Transfer (OT_1^2) proposed by Even, Goldreich, and Lempel in [193], is the most general one. In the

OT_1^2 protocol, there is a Sender with two secrets, S_0 and S_1, and a Receiver with a choice $c \in \{0,1\}$, who tries to learn one and only one of the two secrets. The protocol allows the Receiver to retrieve one and only one of the two secrets based on his/her choice, without the Sender knowing which secret the Receiver had retrieved. An OT_1^2 protocol that uses a public key cryptosystem is presented in [193], and it goes as follows:

1. The Sender selects two public keys (PK_0, SK_0) and (PK_1, SK_1) and sends public parts (PK_0, PK_1) to the Receiver.
2. The Receiver selects a random variable *r*, and based on his/her choice *c*, encrypts *r* with the public key PK_c and sends $PK_c(r)$ to the Sender.
3. The Sender, not knowing what is the choice *c* of the Receiver, decrypts $PK_c(r)$ with each of his/her two private keys and obtains $SK_0(PK_c(r))$ and $SK_1(PK_c(r))$. The Sender then adds S_0 and S_1 to $SK_0[PK_c(r)]$ and $SK_1[(PK_c(r)]$ respectively, and sends $[S_0 + SK_0[PK_c(r)]]$ and $[S_1 + SK_1[PK_c(r)]]$ to the Receiver. The Receiver can then subtract *r* from the received values to learn S_0 or S_1. Table 11 illustrates the complete procedure of OT_1^2.

Table 11. Message sequence for OT_1^2.

Sender		Receiver
Chooses two public keys (PK_0, SK_0) and (PK_1, SK_1)		Chooses a random variable *r*
Sends (PK_0, PK_1)	$\xrightarrow{PK_0, PK_1}$	
	$\xleftarrow{PK_c(r)}$	Chooses $c \in \{0,1\}$. Computes and sends $PK_c(r)$
Computes and sends $[S_0 + SK_0(PK_c(r))]$, $[S_1 + SK_1(PK_c(r))]$	$\xrightarrow{[S_0+SK_0(PK_c(r))],[S_1+SK_1(PK_c(r))]}$	
		Computes $[S_0 + SK_0(PK_c(r))] - r \Rightarrow S_0$ or $[S_1 + SK_1(PK_c(r))] - r \Rightarrow S_1$

Using the RSA [194] public key system with 1024 bits key size, the main communication cost of OT_1^2 is for exchanging the keys and the encrypted random variable *r*, which is approximately 3Kbits(1 Kbits for each of PK_0, PK_1, and $PK_c(r)$). This

information is passed in three exchanged messages between the Sender and Receiver. As for the computational cost, the Receiver is required to encrypt the random value *r* and the Sender is required to decrypt the value twice, once with each key. Based on the benchmark published in [195], we can see that the computational cost can be somewhere around 0.18 *msec* for the Receiver, and 2*4.63 *msec* for the Sender (4.63 *msec* for decrypting $PK_c(r)$ with each of the two keys), if the Sender and Receiver are using Pentium 4 with 2.1 GHz processor, and running Windows XP operating system.

8.5.2 Oblivious polynomial evaluation protocols

An Oblivious Polynomial Evaluation (OPE) protocol [196,197,198] is a protocol between two parties: one party has a polynomial function *P(x)* and the other party has an input x_* and wants to computes $P(x_*)$ in an oblivious way. There are a number of oblivious polynomial evaluation protocols in the literature, each is based on different assumptions: Noar and Pinkas's protocol [196] is based on two assumptions: the existence of a secure oblivious transfer and the intractability of a noisy Polynomial Interpolation. Another protocol proposed by Lindell and Pinkas [197] is based on the assumption that the Decisional Diffie-Hellman Assumption holds also over the group Z_{n^2}, where *n* is the product of two large prime numbers. Recently, Chang *et. al.* [198] proposed a protocol oblivious polynomial evaluation that is based only on Oblivious Transfer. Chang's protocol is very simple, and can be easily modified to support multi-variable polynomials, as well as polynomial over floating-point numbers. We will introduce next Chang's protocol, which we have adopted in this chapter for oblivious polynomial evaluation.

Chang's OPE protocol assumes that the two parties, Alice and Bob, have already agreed that the polynomial is over a field *F*, with a degree at most *d*. Let [*n*] denote the set {1,…,*n*} for any positive integer *n*, and let $m = \lceil \log_2 |F| \rceil$. Assume that Alice is the party with the polynomial $P(x) = \sum_{i=0}^{d} a_i x_i$, Bob has x_* and wants to compute $P(x_*)$. The main idea of the protocol is for Bob to send to Alice the value of $(x_* + r)$, where *r* is some random value, so that Alice cannot find the real value of x_*. Alice would compute the value of $P(x_* + r)$ and send the result to Bob, who can take out the effect of *r*. At the beginning of the protocol, Alice represents each coefficient a_i of P(*x*) as

$a_i = \sum_{j\in[m]} a_{ij} 2^{j-1}$ with $a_{ij} \in \{0,1\}$, while Bob prepares $v_{ij} = 2^{j-1} x_*^i$, for $i \in [d]$, and $j \in [m]$. The rest of the protocol is shown in Table 12.

Table 12. Chang's OPE

Bob		**Alice**
Prepares d_*m pairs $(r_{ij}, v_{ij}+r_{ij})$, with each r_{ij} chosen randomly		$P(x) = \sum_{i=0}^{d} a_i x_i$ $a_i = \sum_{j\in[m]} a_{ij} 2^{j-1}$
	$\xleftarrow{OT_1^2}$ $\xleftarrow{OT_1^2}$ $\vdots$ $\xleftarrow{OT_1^2}$	Runs $i*j$ times OT_1^2, one time for each pair $(r_{ij}, v_{ij}+r_{ij})$, and retrieve r_{ij} if $a_{ij} = 0$ and $v_{ij} + r_{ij}$ otherwise.
	$a_0 + \sum_{d*m}(r_{ij} \mid v_{ij} + r_{ij})$ $\longleftarrow$	Sends to Bob the sum of a_0 and these d_*m values she got
Computes $P(x_*) = a_0 + \sum_{d*m}(r_{ij} \mid r_{ij} + v_{ij}) - \sum_{i,j} r_{ij}$		

From Table 12, we can see clearly that the complexity of the OPE depends mainly on the complexity of the OT_1^2, and since the OT_1^2 is called *d*m* times, we can deduct that the communication and computation complexity of the OPE is *d*m* times the communication and computation complexity of OT_1^2; this means a total of 3* *d*m* Kbits in data exchanged in (3* *d*m*+1) messages, and 0.18**d*m msec* computational cost for the Receiver, compared to 4.63**d*m msec* computational cost for the Sender.

8.5.3 Privacy Preserving Negotiation Protocol

As we mentioned earlier, Oblivious Polynomial Evaluation has been used in a number of interesting applications [196,197,199,200] including privacy preserving data mining, comparing information without leaking it, the list intersection problem, oblivious neural learning, and anonymous initialization for metering, to list a few. In this section, we will show how OPE can be used to solve the problem of private negotiation of QoS parameters, using an example of establishing a communication session between Alice and Bob. In the example, Bob will try to figure out the QoS parameter value that maximizes

his satisfaction, without exceeding the bandwidth limit set by Alice; Bob must also use OPE to compute secretly Alice's satisfaction value with the selected parameter, without passing the parameter to her. If Alice's satisfaction is equal or lower than zero, Bob will try to select another parameter. The procedure is repeated until either Bob finds a parameter that is acceptable to Alice and passes only that parameter to Alice, or until one of them decides to abort. We will run the procedure with one configuration parameter, and since the used OPE protocol is extendible to multi-variable function evaluation, the same procedure could be used when the users' satisfactions are multi-variable functions.

As we mentioned earlier, we will assume that each user has a separate satisfaction function for sending and receiving. We will denote the sending satisfaction function for user A as $S_A(x)$, where x is the user's preference variable, and the corresponding receiving satisfaction function with $R_A(x)$. We will assume that the users' sending satisfaction functions are polynomial functions. Our solution requires that the data sender sends to the receiver the amount of allocated bandwidth that he can allocate for sending the media data. Within the SIP protocol [49] for IP telephony, the allocated bandwidth and other configuration parameters are currently passed using the Session Description Protocol (SDP) [59] or its new version, SDPng [61].

The main idea of our protocol is for the receiver to find the value for the QoS configuration parameter x_* that maximizes his/her satisfaction function $R(x)$, without exceeding the amount of bandwidth allocated by the sender, and without causing his counterpart's sending satisfaction $S_B(x_*)$ to be zero. The receiver would use the OPE protocol, presented above, to compute his/her counterpart's satisfaction value for the selected parameter x_* without revealing the actual value of x_*. The receiver can run several iterations of the protocol, until he/she finds a value that can suit his/her preferences, and which is also accepted by the sender. The pseudo-code of the protocol, with Alice as a data sender and Bob as a data receiver, is as follow:

1. Alice selects $S_{Alice}(x)$, and Bob selects $R_{Bob}(x)$.
2. Alice sends to Bob the amount of allocated bandwidth b.
3. Bob defines $i = 0$, and creates a constraints set, $Cset = Ø$.
4. Bob finds the value x_{*i} that maximizes $R_{Bob}(x)$, subject to the constraints in *Cset and to the constraint that* Bandwidth(x_{*i}) < b

5. Bob uses the OPE to compute $S_{Alice}(x_i)$
6. If $S_{Alice}(x_{*i}) = 0$, then
    a. If $i >$ (allowable number of iterations, *Max_it*) then Abort
    b. Bob adds $(x \neq x_{*i})$ to the set of constraints *Cset*,
    c. $i = i + 1$
    d. Go to Step 4
7. Else $(S_{Alice}(x_{*i}) \neq 0)$ then Bob can pass only the value x_{*i} to Alice. This value x_{*i} can maximize Bob's satisfaction, subject to the constraints that $S_{Alice}(x_{*i}) \neq 0$ and Bandwidth$(x_{*i}) < b$.

If the algorithm exits through Step 7, Bob would have found a value for the variable x* which maximizes his receiving satisfaction value, and without causing the sending satisfaction function of Alice to drop to zero. Bob would pass only that value to Alice, assured that the rest of the preference values are still private.

8.6 Complexity and efficiency

We now examine the communication and computational complexity of the proposed protocol. We will assume that Bob, after *t* iterations, was able to find a value x_0 that maximize his satisfaction and is acceptable by Alice.

As we can see from the protocol in Section 8.5, the computational complexity of the protocol is *t* times that of the OPE, which is in turn *d*m* times that of the OT_1^2. In other words, after *t* iterations, the protocol would cost Alice and Bob *t* d*m**2*4.63 *msec* and *t* d*m**0.18 *msec* respectively. As for the communication complexity, the cost would be somewhere around *t*d*m**3 Kbits exchanged in *t**(3* *d*m*+1) messages.

By looking again at the protocol, we can observe that the complexity of the protocol comes mainly from the exchange of keys, and from encryption and decryption of the random variables R_{ij} during each iteration of the OPE. A simple way to reduce the cost would be to use the same keys random variables for all OPE iterations. Doing this reduces the complexity of the protocol to (*d*m*+2) Kbits communication cost with only (2**t* + 2) messages, and *d*m**2*2.46 *msec* computation cost for Bob and *d*m**0.18 *msec* for Alice. Table 13 shows a compressed version of the protocol.

Table 13. Compressed version of the privacy preserving negotiation protocol

Bob		Alice
Chooses two public keys (PK_0, SK_0) and (PK_1, SK_1)		$P(x) = \sum_{i=0}^{d} a_i x_i$ $a_i = \sum_{j \in [m]} a_{ij} 2^{j-1}$
		Chooses $d*m$ random variables R_{ij} with $1 \le i \le d$; $1 \le j \le m$
Sends (PK_0, PK_1)	$\xrightarrow{PK_0, PK_1}$	
	$\xleftarrow{PK_{a_{ij}}(R_{ij});1 \le i \le d;1 \le j \le m}$	Encrypts R_{ij} with PK_0 if $a_{ij}=0$, else Encrypt R_{ij} with PK_1
Decrypts each $PK_{a_{ij}}(R_{ij})$ with PK_0 and PK_1		
$i = 0$, $Cset = \emptyset$		
Repeat		
$i = i + 1$;		
Computes x_i		
Prepares $d*m$ pairs (r_{ij} , $v_{ij}+r_{ij}$), with each r_{ij} chosen randomly and $v_{ij} = 2^{j-1} x_{*c}^{i}$		
Computes and sends $[r_{ij} + SK_0(PK_{aij}(R_{ij}))]$, $[v_{ij}+r_{ij}+SK_1(PK_{aij}(R_{ij}))]$, with $1 \le i \le d$; $1 \le j \le m$	$\xrightarrow{[r_{ij}+SK_0(PK_{a_{ij}}(R_{ij}))],[r_{ij}+v_{ij}+SK_1(PK_{a_{ij}}(R_{ij}))]; 1 \le i \le d;1 \le j \le m}$	
Computes $P(x_*) = a_0 + \sum_{d*m}(r_{ij} \mid r_{ij} + v_{ij}) - \sum_{i,j} r_{ij}$	$\xleftarrow{a_0 + \sum_{d*m}(r_{ij} \mid v_{ij} + r_{ij})}$	Computes $a_0 + \sum_{d*m}(r_{ij} \mid v_{ij} + r_{ij})$
Until (($P(x^*) \neq 0$) or ($i > Max\_it$))		

8.7 Conclusion and discussion

The concept of privacy in multimedia communication is a new and interesting research area, which has many challenges. While privacy with traditional communication services were provided by trusted telecom provider, the situation today is different, where the insecure Internet has become a major communication network. Current Internet telecommunication protocols do not address the issue of protecting the user's privacy, and addressing the issue is thus becoming a key factor for determining the future of multimedia communications.

In this chapter, we have addressed the problem of protecting the privacy of user preferences during the setup of multimedia communication sessions. In particular, we have shown that the user's privacy can be breached if there is no restriction on what is exchanged during session initiation. We have proposed three solutions with different assumptions and requirements. We have shown how, with some additional computation and communication cost, two communicating parties, using secure distributed computation primitives, may be able to select common preferences without revealing their personal preferences.

We should mention here though that protecting the user's privacy is not for the mass population, since privacy protection protocols, as we have seen, can have a high computational cost on the communication session in terms of additional setup delay. Enabling privacy protection for the user's preferences in multimedia application can have though the potential of providing users with more flexibility and potentially better service, which would come usually at a certain price. Similarly to providing security for communication protocols, where there is inherent tradeoff in realizing performance vs. security in communications [201,202], the additional cost for protecting the user's privacy will be acceptable though if it results in an increase in the overall satisfaction with the task invocation.

Additionally, protecting user's preferences during negotiation can be used though as QoS metric that differentiate between different service providers. Privacy can also be used as a quality of service dimension, and not as a binary dimension: either you have privacy or not. There are already some early works on using security as a QoS dimension and have integrated it into benefit functions [203,204], with a range from minimum to

ideal and similar thing can happen to privacy: Lin [205] used a Quality of Protection parameter to manage the level of protection that the underlying security mechanism provides to each message communication stream. Abdullah et. al. [206] investigated security as a QoS routing issue. Privacy can also follow the same spec as security, and providing the option to protect the privacy of preferences during communication might be valuable for certain users who might be ready to incur extra cost or setup delay to get a better protection.

Chapter 9

Conclusion and Future Work

Telecommunication is all about one thing: connecting people. With many devices available to the user, such as cellular phone, pager, desktop or mobile computer, reaching a person at the right place, over the right device, and using the right data format has become a major challenge. People still have to do the linking between the person and the device to call in order to contact the person; additionally, people still need to try different places before reaching a person, and they still also have to try different configurations for the communication session before finding the configuration that best suit their preferences. This situation imposes a set of new challenges:

1. Tailoring the "connection": basically allowing the user to modify the communication session to best suit his/her preferences, as to when, where, and from who to receive calls, which device to use and with what configuration parameters.
2. Accommodation of many types of networks, and the need to convert between different control protocols and media types.
3. Ubiquitous reachability: enabling people to be reached through publicly available services, and not to restrict communication to go through a static set of devices.
4. Maintaining user privacy: allowing users to negotiate session configuration parameters without jeopardizing their privacy.

Personal mobility is a solution for people who possess multiple access points and who want to be reached regardless of their terminals and location. In this thesis, we have examined personal mobility in relation with selection of communication device, multimedia QoS requirements and parameters configuration. We have proposed an architecture, called MobInTel that can support different types of terminals and network connectivity. The architecture uses our framework for selecting the QoS configuration parameter for the communication session, based on the user preferences, the context of

the user, the available variants of the content, the availability and capabilities of the devices, network connectivity, and the available intermediate adaptation services.

We have also used the framework to develop an agent-based architecture that brings personal and service mobility to the ubiquitous computing environment. A software agent, running on a portable device carried by the user, leverages the existing service discovery protocols to learn about all services available in the vicinity of the user. Short-range wireless technology such as Bluetooth can be used to build a personal area network connecting only devices that are close enough to the user. The proposed architecture supports also service mobility or service hand-off to recompense for service volatility during user movement. Service mobility is required when the user moves away from one service and gets closer to another similar service. The work has focused on building a system where services follow the user as he moves from one place to the other.

We have also used the framework to accommodate the heterogeneity of different receivers in multimedia multicast applications. Customizing the content to accommodate the heterogeneity of receivers is not a trivial task, but we were able to show how user preferences and device capabilities can be used to select the number of channels, as well as the configuration for each channel.

This thesis makes the following contributions:

1. A number of elements contribute to the overall quality of a multimedia application. Our framework identifies these elements and shows how they can be merged together to provide the best possible experience for the user. A major part of the framework is the QoS selection algorithm that selects a chain of intermediate transcoding services that can adapt the content to best suit the requirement of the user and communication session.
2. Given the user profiles and the number of streams to be transmitted, we have shown how a content provider can select the QoS parameters of the stream variants that maximize the average user satisfaction for all the receivers.
3. We have also designed and implemented an architecture for supporting personal and service mobility. The architecture was extended to support these services in ubiquitous computing environments.

4. While current VoIP signaling protocols allow for negotiation of multimedia session configuration parameters, none of these protocols have addressed the issue of protecting the privacy of user preferences. We have shown how distributed secure computation can be used to carry out the negotiation with minimal privacy invasion.

There are still though some open issues that require further research.

1. In our current implementation, users have to explicitly define their preference in their profiles. It would be more interesting to have a system that can derive the user's preferences through analysis of the user's behavior in time. The system should provide a good initial experience and learn quickly from new users, and at the same time, adapt quickly and properly to the change in the user's interests. Work in Artificial Intelligent (AI) or machine learning might be useful here. Another approach might be to using other user's profiles, recommendation or reputation to build and update other users' profiles.
2. Digital Rights Management is now hot topic among content owners, which tries to provide full protection for digital content, including copying and re-using content. It would be very interesting to address the question of how far can we go in adapting original content for network distribution and delivery without violating the rights of the content creator or provider? MPEG 21 has already addressed some but not all of the issues with the proposed MPEG-21 Rights Data Dictionary and Right Expression Language [207].
3. Another interesting research topic would be to investigate the use of grid or Peer-to-Peer computing to carry out the adaptation services. The benefit of grid computing is that users can join and leave the grid at any time, and most importantly, they can add their services to the pool of services available on the grid. The drawback of this approach is that there will not be a central server to carry out the algorithm, since there will not be a central information repository about all network transcoding services. This would require using a peer-to-peer technology to find the best intermediate services that can be used to transcode the content in the network. This approach might also be beneficial to reduce the exponential running time of the QoS selection algorithm.

In addition to these research topics, there are also some open issues with the prototype that we worthwhile investigating and improving upon, including:

1. Stability Study: To continuously search for new services, we currently use a freeware called SDPtool. We noticed in the current implementation, however, that the SDPtool is not stable at all. Sometime it hangs up while trying to browse for new services or fails to find a new service. We do not know what is exactly going on inside the tool since the SDPtool is freeware with no support and bad documentation.
2. Authentication: One of the issues that come to our mind next is the issue of user authentication and authorization, where a user should be authenticated and authorized to use a service in the ubiquitous computing environment. The normal authentication approach such as password protection does not fit properly here, since these services are ubiquitous and they do not know all possible users. An approach similar to [208] where a user receives a token for the service from a trust agent of the service, and the user just presents the token to the service. Additionally, it might be possible not to use any authentication at all, but to use a form of prepayment cards, in cases where the service is very cheap or publicly available. Privacy invasion might be a concern here too.
3. Billing: Another important issue here is the issue of billing the user for the service. Having a billing service in place would encourage the deployment of ubiquitous services. A possible approach is to provide the service with a credit card number to bill for the service usage; but then it becomes of question of how much to trust the service with the credit card information. Also, it is not easy to check whether the user has been billed for a legitimate bill. Using just a log to keep track of these bills might be a solution where the system logs all the transactions and allows the user to review the log later might be a solution (can we use the credit card bill for this purpose?)
4. Usability Study: Now that we have a prototype of the system, it would be really useful to have some usability study to see whether users do really find it useful to have personal and service mobility in ubiquitous computing environments. Specifically, it would be nice to evaluate audio vs. video switching, and whether

automatic switching is preferable over user initiated switching. In particular, we can look at the following issues:

a. Audio Switching: since all users are highly sensitive to audio, it might be that special attention must be paid to audio switching, and even try to minimize the switching. Options such as using the audio on the PDA will eliminate audio switching completely, but this requires the PDA to hold 802.11 connectivity continuously. Other option would be to keep sending the audio data to the fading audio play-out service while the system is switching to another audio service (we can keep sending the data for one minute, let us say),
b. Video Switching: As for audio switching, we think the issue would be to affirm that video switching does not have to be switched that fast, since the video carries a lot of redundancy, and users might accept a black-out for a short time (it will be similar to looking away from the TV while watching something else, for some time). We think people can accept video gaps longer than audio gaps.
c. Using a "**hold"** key to switch a service: Our current implementation supports the use of a special key which, when pressed on the PDA, puts the session on hold, and when pressed again in a new space, the session would continue from the same point where it was stopped. (Such a service would be useful in cases when the user is moving between the office and the living room or between the car and the office). It would be nice to find whether people find using the "hold" better or worse than allowing the system to automatically move the service when the user moves to another location. It would be also interesting to see what effect this would have also on the communicating partner.

Reference

1. Communication and Computing for Distributed Multimedia Systems, Guojun Lu.
2. S. Ramanathan and P. V. Rangan, “Continuous Media Synchronization in Distributed Multimedia Systems,” in Proceedings of the Third International Workshop on Network and Operating Systems Support for Digital Video and Audio, San Diego, November 12-13 1992.
3. L. Lamont, L. Li, R. Brimont, and N. D. Georganas, “Synchronization of Multimedia Data for a Multimedia News-on-Demand Application,” IEEE Journal on Selected Areas in Communications 14(1): 264-278 (1996)
4. W. Tawbi and E. Horlait and J. Stefani, “Video Compression Standards and Quality of Service,” The computer Journal, vol. 36, No. 1, 1993.
5. L.A. Rowe, D.A. Berger, J.E. Baldeschwieler, " The Berkeley Distributed Video-on-Demand System," Multimedia Computing - Proceedings of the Sixth NEC Research Symposium, Ed. T. Ishiguro, SIAM, 1996.
6. G. Miller, G. Baber, and M. Gilliland, “News On-Demand for multimedia Networks,” in Proceedings of the ACM Multimedia 93, ed. P. Venkat Rangan, pp. 383-392, Anaheim (1993).
7. P. Venkat Rangan, “Video conferencing, file storage, and management in multimedia computer systems,” Computer Network and ISDN Systems, 25, pp.901-919, North Holland (1993).
8. M. Altenhofen et al., “The BERKOM multimedia collaboration service,” in Proceedings of the ACM Multimedia 93, ed. P. Venkat Rangan, pp. 457-464, Anaheim.
9. R. Liscano, R.r Impey, Y. Qinxin, and A. Suhayya. “Integrating Multi-Modal Messages across Heterogeneous Network,” in Proceedings of the IEEE International Conference on Communications, June 1997.
10. G. Peersman, S.R. Cvetkovic, C. Smythe, H. Spear and P. Griffiths P, “The Integration of SMS with Voice Based Technologies,” IEE Symposium, Digest No: 1997/147, June 1997, pp.9/1-9/7.

11. A. Vogel, B. Kerherve, G. v. Bochmann, and J. Gecsei, “Distributed Multimedia and QoS: a Survey,” IEEE Multimedia, vol. 2, no.2, pp.10-18, Summer 1995.
12. IETF, “FYI on "What is the Internet?",“ RFC 1462, May 1993.
13. D. Clark, “The Design Philosophy of the DARPA Internet Protocols,” ACM SIGCOMM’88, August 1988.
14. http:\ \www.source.com. Accessed on Jan. 2003.
15. IETF, “Integrated Services in the Internet Architecture: an Overview,” RFC 1633, Jun. 1994.
16. IETF, “Resource ReSerVation Protocol (RSVP) ,” RFC 2208, Sept. 1997
17. L. Mathy, C. Edwards, and D. Hutchison, "The Internet: A Global Telecommunications Solution?," IEEE Network, July/August 2000, Vol.14, No.4, pp 46-57.
18. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, Dec. 1998.
19. C. Rosen, A. Viswanathan, and R. Callon, “MultiProtocol Label Switching Architecture,” IETF, RFC 3031.
20. Y. Rekhter, B. Davie, D. Katz, E. Rosen, G. Swallow, “Tag Switching Architecture”, http://www.cisco.com/warp/public/732/tag/tagsw_ov.htm, April 1997.
21. L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, “LDP Specification,” IETF, RFC 3036.
22. J. Moy, “OSPF version 2,” Internet Engineering Task Force, Request for Comments (Proposed Standard) 2328, Apr. 1998.
23. Y. Rekhter and T. Li, “A Border Gateway Protocol 4 (BGP-4),” RFC 1771, March, 1995.
24. C. Rosen, Y. Rekhter, D. Tappan, D. Farinacci, G. Fedorkow, T. Li, and A. Conta, “MPLS Label Stack Encoding,” IETF, RFC 3032.
25. R. Callon, P.Doolan, N. Feldman, A. Fredette, G. Swallow, A. Viswanathan, “A Framework for MultiProtocol Label Switching,” http://www.ietf.org/internet-drafts/draft-ietf-mpls-framework-02.txt, November 1997.

26. Z. Wang and J. Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications," IEEE Journal on Selected Areas in Communications, vol. 14, n. 7, pp. 1228-1234, 1996.
27. M.R. Garey and D.S. Johnson, “Computers and Intractability- A guide to the Theory of NP-Completeness,” San Fransisco, CA: Preeman, 1979.
28. J. Jaffe, “Algorithms for finding paths with multiple constraints,” Networks, vol. 14, pp. 95-116, 1984.
29. D. S. Reeves and H. F. Salama “A distributed algorithm for delay-constrained unicast routing,” IEEE/ACM Trans. Netw. 8(2): 239-250 (2000)
30. V. Witana, M. Fry, and M. Antoniades, "A Software Framework for Application Level QoS Management," in Proceedings of Seventh International Workshop on Quality of Service, May 1999. http://citeseer.ist.psu.edu/witana99software.html.
31. D. Drucker, “Formal” QoS Not Yet A Priority, InterNetWeek, June 18, 2001.
32. A. Hafid and G. v. Bochmann, “Quality of service adaptation in distributed multimedia applications,” Multimedia Systems Journal (ACM), Vol. 6, No. 5 (1998), pp. 299-315.
33. M. Fry, A. Seneviratne, A. Vogel and V Witana, “Delivering QoS Controlled Continuous Media on the World Wide Web,” In Proc IWQoS, pages 115-124, 1996.
34. K. Lakshman and R. Yavatkar. AQUA, “An Adaptive End-System Quality of Service Architecture,” High-Speed Networking for Multimedia Applications. Nov. 1996.
35. P. Bellavista, A. Corradi, and C. Stefanelli “Application-Level QoS Control for Video-on-Demand,” IEEE Internet Computing 7(6): 16-24 (2003)
36. G. Ghinea and J.P. Thomas, "QoS Impact on User Perception and Understanding of Multimedia Video Clips," Proceedings of ACM Multimedia ’98, Bristol, United Kingdom, 1998
37. RACE IBC CFS D5510, “General Aspects of Quality of Service and System Performance in IBC,” RIC, Brussels, 91.

38. S. Fischer, and R. Keller, “Quality of Service Mapping in Distributed Multimedia Systems,” IEEE International Conference on Multimedia Networking (MmNet95) (), pp. 132-141 , 1995.
39. M. Alfano, and R. Sigle “Controlling QoS in a Collaborative Multimedia Environment,” Proceedings of the 5th IEEE International Symposium on High Performance Distributed Computing (HPDC 5), New York, 1996.
40. A. Vogel, G. v. Bochmann, R. Dssouli, J. Gecsei, A. Hafid and B. Kerherve, "On QoS Negotiation in Distributed Multimedia Application," Proc. Protocol for High Speed Networks, April 1994.
41. A. Richards, G. Rogers, V. Witana, and M. Antoniades, “Mapping user level QoS from a single parameter,” In Second IFIP/IEEE International Conference on Management of Multimedia Networks and Services, Versailles, November 1998.
42. E. W. Fulp, M. Ott, D. Reininger, and D. S. Reeves, “Paying for QoS: An Optimal Distributed Algorithm for Pricing Network Resources," Proc. of International Workshop on Quality of Service (IWQOS '98), May 1998, pp. 75--84.
43. R. Cocchi, D. Estrin, S. Shenker, and L. Zhang, "A study of priority pricing in multiple service class networks," in ACM SIGCOMM, 1992.
44. N. Bhatti, A. Bouch, and A.J. Kuchinsky, “Integrating user-perceived quality into Web server design,” Proceedings of WWW'00. May 15-19th, 2000, Amsterdam, Netherlands, pp 1-16.
45. M. Mcllhagga, A. Light, I. Wakeman, “Towards a Design Methodology for Adaptive Applications,” MOBICOM 1998: 133-144
46. A. Hafid, G.v.Bochmann and R.Dssouli, “A Negotiation Approach with Future Reservation (NAFUR): A Detailed Study,” Computer Networks and ISDN Journal, 1996.
47. IETF, “TRANSMISSION CONTROL PROTOCOL”, Sept. 1981.
48. ITU-T Recommendation H.323, "Packet-Based Multimedia Communications Systems," February 1998.
49. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, June 2002.

50. ITU-T Recommendation H.225, ”Call signaling protocols and media stream packetization for packet-based multimedia communication systems,” 2000.
51. ITU-T Recommendation H.245 (1998) "Control Protocol for Multimedia Communication."
52. ITU-T Recommendation H.235, “Security and encryption for H Series (H.323 and other H.245 based) multimedia terminals”, 1998.
53. ITU-T Recommendation H.332: "H.323 Extended for Loosely-coupled Conferences".
54. I. Dalgic and H. Fang, “Comparison of H.323 and SIP for IP Telephony Signaling,” Proc. of Photonics East, Boston, Massachusetts, September 20-22, 1999
55. H. Schulrzrinne, J. Rosenberg, “A Comparison of SIP and H.323 for Internet Telephony,” Network and Operating System Support for Digital Audio and Video (NOSSDAV), Cambridge, England, July 1998.
56. IETF, “User Datagram Protocol”, RFC 768, Aug. 1980.
57. J. Rosenberg and H. Schulzrinne, “Models for multi party conferencing in SIP,” Internet Draft, Internet Engineering Task Force, Nov. 2000.
58. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003.
59. M. Handley and V. Jacobson, “SDP: Session Description Protocol,” RFC 2327, April 1998.
60. Lawrence Berkeley Labs, http://www.lbl.gov/
61. D. Kutscher, J. Ott, and C. Bormann, “Session Description and Capability Negotiation,” Internet Draft draft-ietf-mmusic-sdpng-07.txt, Work in Progress; October 2003.
62. M. Handley, C. Perkins C. and E. Whelan, “SAP: Session Announcement Protocol,” RFC 2974, Oct. 2000.
63. C. Groves, M. Pantaleo, T. Anderson and T. Taylor, ”The Megaco/H.248 Gateway Control Protocol, version 2,” http://ietf.org/internet-drafts/draft-ietf-megaco-h248v2-04.txt, Work in Progress.
64. Yahoo. htpp://www.yahoo.com

65. eBAY. htpp://www.e-bay.com
66. S. Chandra and C.S. Ellis, “JPEG Compression Metric as a Quality Aware Image Transcoding,” in Second Usenix Symposium on Internet Technologies and Systems (USITS '99), Oct 12, 1999
67. R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas, “Dynamic adaptation in an image trans-coding proxy for mobile WWW browsing,” IEEE Personal Communication, vol. 5 (6), Dec. 1998.
68. Cisco CallManager. http://www.cisco.ca
69. J. R. Smith, R. Mohan, C.-S. Li, “Scalable Multimedia Delivery for Pervasive Computing,” ACM Multimedia '99, Oct. 30 - Nov. 5, 1999.
70. K. Nahrstedt, J. Smith, The QoS Broker, IEEE Multimedia, Vol. 2, No. 1, 1995, pp. 53-67.
71. B. Field, T. Znati, and D. Mosse, “V-net: A Framework for a Versatile Network Architecture to Support Real-Time Communication Performance Guarantees,” InfoComm, 1995.
72 . Z. Morley Mao, H. Wilson So, B. Kang, and R. H. Katz, “Network Support for Mobile Multimedia using a Self-adaptive Distributed Proxy,” 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV-2001).
73. R. Procter, M. Hartswood, A. McKinlay, S. Gallacher, “An investigation of the influence of network quality of service on the effectiveness of multimedia communication,” Group 99, ACM Press (1999), 160-168.
74. J. Gowan and J. Downs, “Video conferencing human-machine interface: A field study,” Information and Management, 27 (1994), 341-356.
75. I. Busse, B. Deffner, and H. Schulzrinne, “Dynamic QoS control of multimedia applications based on RTP,” Computer Communications, 19, 1996, pp. 49-58.
76. A. Bouch, A. Kuchinsky, and N. Bhatti, “Quality is in the eye of the beholder: Meeting users’ requirements for internet quality of service,” Proc. CHI 2000, CHI Letters 2(1), 297-304.

77. R. Apteker, J. Fisher, V. Kisimov, and H. Neishlos, “Video acceptability and frame rate,” IEEE Multimedia, 2, 3 (1995), 32-40.
78. A. Fox, S.D. Gribble, and Y. Chawathe, “Adapting to Network and Client Variation Using Infrastructural Proxies: Lessons and Perspectives,” IEEE Personal Communications, August 1998.
79. W.Y. Lum and F.C.M. Lau, "On Balancing Between Trans-coding Overhead and Spatial Consumption in Content Adaptation," Mobicom 2002, Atlanta, USA, September 2002, 239-250.
80. C. Y. Chang and M. S. Chen “Exploring Aggregate Effect with Weighted Transcoding Graphs for Efficient Cache Replacement in Transcoding Proxies,” in Proceedings of the 18th IEEE International Conference on Data Engineering (ICDE-O), 2002.
81. Z. Lei and N.D. Georganas, "Context-based Media Adaptation in Pervasive Computing," Proc. Can.Conf. on Electr. and Comp. Engg (CCECE'2001), Toronto, May 2001
82. A. Hafid and G.v. Bochmann, “Quality of Service Negotiation in News-on-Demand Systems: an Implementation,” In Proceedings of the Third International Workshop on Protocols for Multimedia Systems, Madrid, Spain, 1996
83. R.Mohan, J.R. Smith and C.S. Li, “Adapting Multimedia Internet Content for Universal Access,” IEEE Trans. on Multimedia, 1(1):104–114, 1999.
84. S. Björk, L.E. Holmquist, J. Redström, I. Bretan, R. Danielsson, J. Karlgren, and K. Franzén, “WEST: a Web browser for small terminals,” Proceedings of the 12th annual ACM symposium on User interface software and technology, p.187-196, November 07-10, 1999, Asheville, North Carolina, United States
85. B. Fisher, G. Agelidis, J. Dill, P. Tan, G. Collaud, and C. Jones, "CZWeb: Fish-Eye Views for Visualizing the World-Wide Web," Proc. of the 7th Int. Conf. on Human-Computer Interaction (HCI International '97), 719-722, 1997.
86. S. Chandra, C. Ellis, and A. Vahdat, “Application-Level Differentiated Multimedia Web Services Using Quality Aware Transcoding,” IEEE Journal on Selected Areas in Communications, 2000.

87. R. Floyd and B. Housel, “Mobile Web Access Using eNetwork Web Express,” IEEE Personal Communications, 5(5):47–52, 1998.
88. A. Fox, S.D. Gribble, Y. Chawathe, E.A. Brewer, and P. Gauthier, “Cluster-Based Scalable Network Services,” in Proc. 16th ACM Symp. On Operating Systems Principles, pages 78–91, Saint-Malo, France, 1997.
89. http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm
90. http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm
91. Ubiquitous Computing (Xerox PARC) http://www.ubiq.com/hypertext/wiser/UbiHome.html
92. Contextual Computing Group (Georgia Tech). http://www.cc.gatech.edu/ccg/
93. Ambiante (Collaborative Buildings) (GMD) www.darmstadt.gmd.de/ambiente
94. A.K. Dey, “Providing architectural Support for Building Context-Aware Applications,” PhD thesis, College of Computing, Georgia Institute of Technology, December 2000.
95. A. Schmidt, M. Beigl, and H. Gellersen, “There is More to Context than Location”, Computers & Graphics, vol. 23, no. 6, Dec. 1999, pp. 893-902; http://www.elsevier.com.
96. Wireless Application Forum, http://www.wapforum.org/
97. C.W. Ng, P.Y. Tan, and H. Cheng, "Quality of Service Extension to IRML,” IETF INTERNET-DRAFT, 'draft-ng-opes-irmlqos-00.txt', July 2001.
98. M. Katchabaw, H. Lutfiyya, and M. Bauer, “Driving resource management with application-level quality of service specifications,” ICE 98, ACM Press (1998), 83-91.
99. C. Poellabauer, H. Abbasi, and K. Schwan, “Cooperative run-time management of adaptive applications and distributed resources,” Proc. Multimedia ’02, ACM Press (2002).
100. D. Wu., Y.T. Hou, and Y. Zhang, "Scalable Video Coding and Transport over Broad-band Wireless Networks," Proc. IEEE, vol. 89, no. 1, pg 6-20, Jan 2001.
101. S. Hollfelder, A. Kraiss, and T.C. Rakow, “A Client-Controlled Adaptation Framework for Multimedia Database Systems,” in R. Steinmetz and L.C. Wolf,

editors, Proc. IDMS'97, pp 397-409, Darmstadt, Germany, September 10-12, Springer 1997.

102. JINI (TM): Http: //java.sun.com/product/JINI/. 1998.
103. E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol. Version 2.," http://ietf.org/rfc/rfc2608.txt.
104. W3C. WSDL: Web Service Description Language, http://www.w3.org/TR/wsdl, 2002.
105. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. "Introduction to algorithms." MIT Press and McGraw-Hill Book Company, 6th edition, 1992.
106. JINI (TM): Http: //java.sun.com/product/JINI/. 1998.
107. E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol. Version 2.," http://ietf.org/rfc/rfc2608.txt.
108. W3C. WSDL: Web Service Description Language, http://www.w3.org/TR/wsdl, 2002.
109 I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications," Proc. ACM SIGCOMM 2001. San Diego, California, 2001.
110. S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker; A Scalable Content-Addressable Network. SIGCOMM01, August 27-31, 2001, San Diego, California, USA.
111. A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001). Heidelberg, Germany, November 2001.
112. J. Wu, S. Campbell, J. M. Savoie, H. Zhang, G. v. Bochmann and B. St.Arnaud "User-managed end-to-end lightpath provisioning over CA*net 4," in the proceedings of the National Fiber Optic Engineers Conference (NFOEC), Orlando, FL, USA, Sept 7-11, 2003, pp. 275-282.
113. Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," RFC 1771, Mar. 1995.

114. T. Jiang, E. Zegura, M. Ammar, "Inter-Receiver Fair Multicast Communication over the Internet," in Proc. of NOSSDAV 99, June 1999.
115. S. Yan Cheung, Mostafa H. Ammar, and Xue Li , “On the use of destination set grouping to improve fairness in multicast video distribution,” in Proc. of IEEE INFOCOM'96, March 1996.
116. B. J. Vickers, C. Albuquerque and T. Suda, "Source-adaptive multi-layered multicast algorithms for real-time video distribution," in IEEE/ACM Transactions on Networking, 1999.
117. X. R. XU, A.C. Myers, H. Zhang, and R. Yavatkar, “Resilient multicast support for continuous-media applications,” Proceedings of NOSSDAV'97.
118. S. Floyd, V. Jacobson, S. McCanne, C. Liu, and L. Zhang, “A Reliable multicast framework for light-weight sessions and application level framing,” in Proceeding of SIGCOMM, 1995.
119. M. Grossglausser “Optimal deterministic timeouts for reliable scalable multicast,” IEEE Journal on Selected Area in Communications, vol. 15, no.3, 1997.
120. J. Bolot, T. Turletti, and I. Wakeman, “Scalable feedback control for multicast video distribution in the Internet,” in Proc. Of ACM SIGCOMM, pp. 58-67, August 1994.
121. R. Yavatkar, J. Griffoen, and M. Sudan, “A reliable dissemination protocol for interactive collaborative applications,” in Proceedings of INFOCOM, 1991.
122. T. Jiang, M. Ammar, E. Zegura, “Inter-Receiver fairness: A Novel performance measure for Multicast ABR sessions,” in Proceedings of ACM SIGMETRICS'98 , Madison, Wisconsin, June 1998.
123. S. McCanne, V. Jacobson, and M. Vetterli, “Receiver-driven layered multicast,” in Proc. of ACM SIGCOMM, pp. 117-130, August 1996.
124. L. Wu, R. Sharma and B. Smith, "Thin streams: An architecture for multicasting layered video," in Proc. of NOSSDAV'97, St.Louise,Missouri, May 1997.
125. P. Assunção and M. Ghanbari, “Multi-Casting of MPEG-2 Video with Multiple Bandwidth Constraints,” in Proc. of the 7th Int’l Workshop on Packet Video, pp. 235-238, March 1996.

126. G. v. Bochmann and Z. Yang, ”Quality of service management for teleteaching applications using the MPEG-4/DMIF,” in Intern. Workshop on Interactive Distr. Multimedia Systems and Telecom. Services, Toulouse, Oct. 1999.
127. M. R. Garey and D. S. Johnson “Computers and Intractability. A Guide to the Theory of NP-Completeness,” Freeman, San Francisco, 1979.
128. Michael J. Donahoo and Sunila R. Ainapure, "Scalable multicast representative member selection", INFOCOM 2001.
129. D. DeLucia, K. Obraczka, “Multicast feedback suppression using representatives,” in Proceedings of IEEE Infocom'97.
130. C. E. Perkins and P. Bhagwat, “A mobile networking system based on the Internet protocol,” IRRR Personal Communications, vol. 1, pp. 32-41, First Quarter 1994.
131. http://www.sun.com/products/sunray1/hotdesk.html
132. R. Pandya “Emerging mobile and personal communication systems,” IEEE Communications Magazine, vol. 33, pp. 44-52, June 1995.
133. Skype. http/www.skype.com.
134. J. Rosenberg, “Presence: the best thing that ever happened to Voice”, http://www.cconvergence.com/article/CTM20001023S0001, access on 11/05/2000.
135. Mema Roussopoulos, Petros Maniatis, Edward Swierk, Kevin Lai, Guido Appenzeller and Mary Baker, "Person-level Routing in the Mobile People Architecture," Proceedings of the USENIX Symposium on Internet Technologies and Systems, October 1999.
136. H. J. Wang, B. Raman, R. Biswas, C.Chuah, R. Gummadi, B. Hohlt, X. Hong, E.Kiciman, Z.Mao, J. S. Shih, L. Subramanian, B. Y. Zhao, A. D. Joseph, R. H. Katz. “ICEBERG: An Internet-core Network Architecture for Integrated Communications,” IEEE Personal Communications (2000): Special Issue on IP-based Mobile Telecommunication Networks.
137. N. Anerousis, R. Gopalakrishnan, C.R. Kalmanek, A.E. Kaplan, W.T. Marshall, P.P. Mishra, P.Z. Onufryk, K.K. Ramakrishnan, C.J. Sreenan “TOPS: An architecture for telephony over packet networks,” IEEE Journal of Selected Areas in Communications, Jan. 1999.

138. O. Kahane and S. Petrack, “Call Management Agent System: Requirements, Function, Architecture and Protocol,” IMTC Voice over IP Forum Submission VOIP97-010, 1997.
139. H. Schulzrinne and E. Wedlund, "Application-Layer Mobility using SIP," ACM Mobile Computing and Communications Review, Vol. 4, No. 3, July 2000, pp. 47-57.
140. N. Griffeth and H. Velthuijsen. "The Negotiating Agents Approach to Runtime Feature Interaction Resolution". In International Workshop on Feature Interactions in Telecommunications Systems, pages 217-235, 1994.
141. D. Pinard, M. Weiss, T. Gray, Issues In Using an Agent Framework For Converged Voice/Data Applications, Second International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents (PAAM’97), London, UK, 21-23 April 1997.
142. D. Pinard, “Reducing the Feature Interaction Problem in Communication Systems Using an Agent-Based Architecture,” FIW 2003: 13-22.
143. I. Zibman, C. Woolf, P. O'Reilly, L. Strickland, D. Willis, and J. Visser, "An Architectural Approach to Minimizing Feature Interactions in Telecommunications," IEEE/ACM Trans. Networking, vol. 4, no. 4, pp. 582-596, Aug. 1996.
144. Mobilennium – The UMTS Forum Newsletter, No. 5 Nov (1998)
145. DTI, Developing Third Generation Mobile and Personal Communications into the 21st Century, 31 May 1997, http://www.gtnet.gov.uk/radiocom
146. S. Lloyd, R.Hadingham, and A.Pearmain “Virtual Home Environments Negotiated by Agents,” 2nd International ACTS Workshop on Advanced Services in Fixed and Mobile Telecommunications Networks, CWC, Singapore, 1999.
147. http://www.uk.infowin.org/ACTS/RUS/PROJECTS/ac341.htm
148. T.Wierlemann, T.Kassing, B.Kreller “Usability of a Mobile Multi-Media Service Environment for UMTS,” 3rd Acts Mobile Communication Summit, June 8-11,1998, V.1 p151-166.
149. Bluetooth: http://www.bluetooth.com

150. A. Hafid and G. V. Bochmann, “An Approach to Quality of Service Management in Distributed Multimedia Application: Design and an Implementation,” Multimedia Tools & Applications, Vol. 9, No. 2, 1999.
151. I. Dupré la Tour, G.v. Bochmann, and J.-Y. Chouinard, "A Secure Authentication Infrastructure for Mobile Communication Services over the Internet," 6th International Federation for Information Processing (IFIP) Communications and Multimedia conference (CMS'2001), Darmstadt, Germany, pp. 405-416, May 21 to 22, 2001.
152. G.v. Bochmann and E. Zhang, ``A Secure Authentication Infrastructure for Mobile Users," A Book Chapter in Advances in Security and Payment Methods for Mobile Commerce, to be published in 2005
153. S. Petrack and L. Conroy, “The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Services,” RFC 2848, June 2000.
154. http://www.ubiq.com/weiser
155. J. Haartseen, M. Naghshineh, and J. Inouye, “Bluetooth: Vision, Goals, and Architecture,” ACM Mobile Computing and Communications Review, Vol. 2, No. 4, October 1998, pp. 38-45.
156. H. Schulzrinne, “Personal mobility for multimedia services in the Internet,” in European Workshop on Interactive Distributed Multimedia Systems and Services (IDMS), (Berlin, Germany), Mar. 1996.
157. S. Campadello “Dynamic Composition of execution Environment for Adaptive Nomadic Application, “MATA01 Workshop, August 2001, Montreal Canada.
158. M. Barbeau, V. Azondekon, and R. Liscano, “Service Selection in Mobile Computing Based on Proximity Confirmation Using Infrared,” MICON 2002.
159. G. Schneider, C. Hoymann, and S. Goose, “Ad-hoc Personal Ubiquitous Multimedia Services via UPnP,” Proceedings of the IEEE International Conference on Multimedia and Exposition, Tokyo, Japan; Aug. 2001
160. M. Beigl, A. Schmidt, M. Lauff, and H.W. Gellersen, “The UbicompBrowser,” Proceedings of the 4th ERCIM Workshop on User Interfaces for All, October 1998, Stockholm, Sweden

161. Smart Space Laboratory. http://www.ht.sfc.keio.ac.jp/SSLab/
162. N. Kohtake, K. Matsumiya, K. Takashio, and H. Tokuda, “Smart Device Collaboration for Ubiquitous Computing Environment,” Workshop of Multi-Device Interface for Ubiquitous Peripheral Interaction, Fifth International Conference on Ubiquitous Computing (UbiComp2003), USA, 2003.
163. IEEE802.11: http://grouper.ieee.org/groups/802/11/main.html
164. Bluetooth Special Interest Group (SIG), “Service Discovery Protocol,” SIG Specification version 1.0, Volume 1 Core, part E, pp 233-384
165. E. Horvitz, A. Jacobs, and D. Hovel, “Attention-Sensitive Alerting,” in: Proceedings of UAI ‘99, Stockholm, Sweden, July 1999, pp. 305-313. San Francisco: Morgan Kaufmann
166. K. El-Khatib, Zhen E. Zhang, N. Hadibi, and G. v. Bochmann, “Personal and Service Mobility in Ubiquitous Computing Environments", Journal of Wireless communications and Mobile Computing, (Accepted) to appear, 2004.
167. http://sourceforge.net/
168. M. Haase, I. Sedov, S. Preuss, C. Cap, and D. Timmermann, "Time and Energy Efficient Service Discovery in Bluetooth," Procedings of the 57th IEEE Semiannual Vehicular Technology Conference, Jeju, Korea, April 2003.
169. In the Matter of GeoCities, a corporation, FTC File No. 9823015 <http://www.ftc.gov/os/1999/9902/9823015cmp.htm>.
170. J. Borking, “Proposal for Building a Privacy Guardian for the Electronic Age,” in H. Federrath, editor, Anonymity 2000, Volume 2009 of Lecture Notes in Computer Science, Pages 130-140, Springer-Verlag, 2000.
171. R. Hes and J. Borking, “Privacy-Enhancing Technolgies: The Path to Anonymity,” Revised Edition. A&V-11. Den Haag: Registratiekamer, 1998.
172. http://www.maxware.com
173. http://www.anonymizer.com
174. E. Gabber, P. Gibbons, Y. Matias, and A. Mayer, “How to make personalized web browsing simple, secure, and anonymous,” in Proc. of Financial Cryptography '97 (1997).

175. http://w3c.org/p3p
176. http://w3c.org/APPEL
177. S. Tamaru, J. Nakazawa, K. Takashio, H. Tokuda, “PPNP: A Privacy Profile Negotiation Protocol for Services in Public Spaces,” First International Workshop on Ubiquitous Systems for Supporting Social Interaction and Face-to-Face Communication in Public Spaces at the Ubicomp 2003 Conference.
178. A. Rezgui, M. Ouzzani, A. Bouguettaya, and B. Medjahed, “Preserving privacy in web services,” Proceedings of the International workshop on Web Information and Data Management, pp. 56-62, 2002.
179. D. M. Zhang and W. Y. Wong, “A Web-Based Negotiation Agent Using CBR,” PRICAI Workshops 2000, pp. 183-198.
180. G. Yee, L. Korba, “Bilateral E-services Negotiation Under Uncertainty”, Proceedings, the 2003 International Symposium on Applications and the Internet (SAINT2003), Orlando, Florida, Jan. 27-31, 2003.
181. K. E. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis, “Protecting Privacy during On-line Trust Negotiation,” 2nd Workshop on Privacy Enhancing Technologies, San Francisco, CA, April 2002.
182. S. E. Aldrich, R. T. Marks, M. M. Frey, M. A. Goulde, J. M. Lewis, and P. B. Seybold, “What kind of the Total Customer Experience Does Your E-business Deliver?,” Patricia Seybold Group, August 2000.
183. T. M. O’ Neil, “Quality of Experience and Quality of Service, For IP Video Conferencing,” White Paper by Polycom.
184. “Assuring QoE on Next generation Networks,” White Paper by Empririx.
185. M. Siller and JC Woods, “Improving Quality of Experience for Multimedia Services by QoS Arbitration on a QoE Framework,” International Conference on Packet Video, 28-29 April 2003, Nantes, France
186. L. Alben, “Defining the criteria for effective interaction design: Quality of experience,” Interactions, 3, 3, 11-15. 1996.
187. A. van Moorsel, “Metrics for the internet age: Quality of experience and quality of business,” Technical report no: HPL-2001-179, http://www.hpl.hp.com/techreports

188. B. Bauer, and A.S. Patrick, “A Human Factors Extension to the Seven-Layer OSI Reference Model,” Retrieved January 6, 2004, from http://www.andrewpatrick.ca/OSI/10layer.html
189. M. Nilsson, H. Lindskog, and S. Fischer-Hubner, “Privacy Enhancement in the Mobile Internet,” in Proceedings of the IFIP WG 9.6/11.7 Working Conference on Security and Control of IT in Society, Bratislava, June 15-16, 2001, ISBN 3-901882-13-8.
190. CC/PP: http://www.w3.org/Mobile/CCPP/
191. H. Zandbelt, B. Hulsebosch, H. Eertink, “IDsec: Virtual Identity on the Internet”, Internet Draft ,Telematica Instituut, draft-zandbelt-idsec-01.txt. May 2002.
192 J. Halpern and M.O. Rabin, "A Logic to Reason about likehood", Proceedings of the 15th Annual ACM Symposium on the Theory of Computing, 1983, pp. 310-319.
193. S. Even, O. Goldreich, and A. Lempel, "A randomized Protocol for signing contracts", Communications of the ACM, Vol. 28, No. 6, 1985, pp. 637-647.
194. R. L. Rivest, A. Shamir and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems”, Communications of ACM, Vol.21, No.2, pp.120-126, Feb 1978.
195. http://www.eskimo.com/~weidai/benchmarks.html
196. M. Naor and B. Pinkas, “Oblivious transfer and polynomial evaluation,” in Proc. 31st Ann. ACM Symp. Theory of Computing, 1999, pp. 245–254.
197. Y. Lindell and B. Pinkas, “Privacy preserving data mining,” Proc. CRYPTO 2000, Lecture Notes in Computer Science, Vol. 1880 (Springer, 2000), pp. 36–54.
198. Y.C. Chang, and C.J. Lu, “Oblivious Polynomial Evaluation and Oblivious Neural Learning,” ASIACRYPT 2001: 369-384
199. N. Gilboa, “Two party RSA key generation”, CRYPTO 1999, 116-129.
200. Y. Ishai and E. Kushilevitz, Randomizing Polynomials, “A New Representation with Applications to Round-Efficient Secure Computation”, IEEE Symposium on Foundations of Computer Science, 2000.

201 P.A. Schneck and K. Schwan. “Dynamic Authentication for High-Performance Networked Applications”, Georgia Institute of Technology College of Computing Technical Report GIT-CC-98-08, 1998.

202 T. E. Levin, C. E. Irvine, and E. Spyropoulou, "Quality of Security Service: Adaptive Security", to appear in The Handbook of Information Security, John Wiley & Sons, Inc. December 2005

203 C. Irvine and T. Levin. “Toward Quality of Security Service in a Resource Management System Benefit Function,” Proceedings of the Heterogeneous Computing Workshop. Cancun, Mexico, 2000.

204 J. Kim, D. Hensgen, T. Kidd, H. Siegel, D. St. John, C. Irvine, T. Levin, N. Porter, V. Prasanna, and R. Freund. “A QoS Performance Measure Framework for Distributed Heterogeneous Networks,” in Proceedings of the Eighth Euromicro Workshop on Parallel and Distributed Processing. pages 18-27. Rhodes, Greece. (2000).

205 J. Linn (1993), “Generic Security Service Application Program Interface,” IETF RFC 1508.

206 A. N. Alghannam, M. E. Woodward , J.E. Mellor, “Security As A QoS Routing Issue,” Computing Dept. / University of Bradford

207 MPEG-21 Overview v.5: http://www.chiariglione.org/mpeg/standards/mpeg-21. Accessed on August 2004.

208. L. Kagal, T. Finin and A. Joshi, “TrustBased Security in Pervasive Computing Environments,” in IEEE Computer, December 2001.

Appendix A
User Profile Schema

This appendix presents the schema for the trans-coder used in the prototype.

```
<?xml version="1.0" ?>

<!--
    Document   : user_profile.xsd
    Created on : May 18, 2004, 10:04 PM
    Organization: University of Ottawa,
    Description:
        Definition of a user’s profile
-->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.site.uottawa.ca/school/research/DSRLab"
        xmlns="http://www.site.uottawa.ca/school/research/DSRLab"
        elementFormDefault="qualified">
   <xsd:element name="userProfile">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="sn" type="humanName"/>
            <xsd:element name="cn" type="humanName"/>
            <xsd:element name="employer" type="xsd:token"/>
            <xsd:element name="organizationMembership" type="xsd:token"
maxOccurs="unbounded"/>
            <xsd:element name="dateofBirth" type="xsd:date"/>
            <xsd:element name="postalAddress" type="xsd:string"/>
            <xsd:element name="homepageURL" type="xsd:anyURI"/>
            <xsd:element name="email" maxOccurs="unbounded">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:token">
                     <xsd:pattern value="[\w]+([\-\._]?[\w]+)*@[\w]+([\-
\._]?[\w]+)*\.[\w]+"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:element>
            <xsd:element name="usrID">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                     <xsd:maxLength value="8"/>
                     <xsd:minLength value="3"/>
                     <xsd:pattern value="[a-zA-Z0-9]+"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:element>
            <xsd:element name="userPassword">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:normalizedString">
                     <xsd:minLength value="6"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:element>
            <xsd:element name="QoSWeight">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:decimal">
                     <xsd:totalDigits value='3'/>
                     <xsd:fractionDigits value='1'/>
```

```
                        <xsd:minInclusive value='0'/>
                        <xsd:maxInclusive value='1'/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="timePolicy" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:all>
                        <xsd:element name="beginningTime" type="xsd:time"/>
                        <xsd:element name="endTime" type="xsd:time"/>
                        <xsd:element name="policyID"
type="xsd:nonNegativeInteger"/>
                        <xsd:element name="weekday" type="xsd:boolean"/>
                    </xsd:all>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="caller" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:all>
                        <xsd:element name="callerID"
type="xsd:nonNegativeInteger"/>
                        <xsd:element name="idealResolution" type="resolution"/>
                        <xsd:element name="acceptableResolution"
type="resolution"/>
                        <xsd:element name="idealAudio" type="audioQuality"/>
                        <xsd:element name="acceptableAudio"
type="audioQuality"/>
                        <xsd:element name="acceptableFramerate"
type="framerate"/>
                        <xsd:element name="idealFramerate" type="framerate"/>
                        <xsd:element name="priceCeiling">
                            <xsd:simpleType>
                                <xsd:restriction base="xsd:decimal">
                                    <xsd:fractionDigits value="2"/>
                                    <xsd:minExclusive value="0"/>
                                </xsd:restriction>
                            </xsd:simpleType>
                        </xsd:element>
                        <xsd:element name="cn" type="humanName"/>
                        <xsd:element name="priority">
                            <xsd:simpleType>
                                <xsd:restriction base="xsd:positiveInteger">
                                    <xsd:maxInclusive value="10"/>
                                </xsd:restriction>
                            </xsd:simpleType>
                        </xsd:element>
                        <xsd:element name="audioWeight">
                            <xsd:simpleType>
                                <xsd:restriction base="xsd:nonNegativeInteger">
                                    <xsd:maxInclusive value="10"/>
                                </xsd:restriction>
                            </xsd:simpleType>
                        </xsd:element>
                        <xsd:element name="videoWeight">
                            <xsd:simpleType>
                                <xsd:restriction base="xsd:nonNegativeInteger">
                                    <xsd:maxInclusive value="10"/>
                                </xsd:restriction>
                            </xsd:simpleType>
                        </xsd:element>
                        <xsd:element name="QoSWeight">
                            <xsd:simpleType>
                                <xsd:restriction base="xsd:decimal">
```

```
                                    <xsd:totalDigits value='3'/>
                                    <xsd:fractionDigits value='1'/>
                                    <xsd:minInclusive value='0'/>
                                    <xsd:maxInclusive value='1'/>
                                 </xsd:restriction>
                              </xsd:simpleType>
                           </xsd:element>
                        </xsd:all>
                     </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="deviceInfo" maxOccurs="unbounded">
                     <xsd:complexType>
                        <xsd:all>
                           <xsd:element name="deviceAddress" type="IP"/>
                           <xsd:element name="deviceProfile">
                              <xsd:simpleType>
                                 <xsd:restriction base="xsd:token">
                                    <xsd:pattern
value="deviceName=[^=,]+(,[\w]+=[^=,]+)+"/>
                                 </xsd:restriction>
                              </xsd:simpleType>
                           </xsd:element>
                           <xsd:element name="deviceID"
type="xsd:nonNegativeInteger"/>
                        </xsd:all>
                     </xsd:complexType>
                  </xsd:element>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
         <xsd:simpleType name="humanName">
            <xsd:restriction base="xsd:token">
               <xsd:pattern value="[a-zA-Z]+[a-zA-Z \.\-]*"/>
            </xsd:restriction>
         </xsd:simpleType>
         <xsd:simpleType name="resolution">
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="1024X768"/>
               <xsd:enumeration value="800X600"/>
               <xsd:enumeration value="640X480"/>
               <xsd:enumeration value="320X240"/>
            </xsd:restriction>
         </xsd:simpleType>
         <xsd:simpleType name="audioQuality">
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="CD"/>
               <xsd:enumeration value="FM"/>
               <xsd:enumeration value="Telephone"/>
            </xsd:restriction>
         </xsd:simpleType>
         <xsd:simpleType name="framerate">
            <xsd:restriction base="xsd:positiveInteger">
               <xsd:maxInclusive value="30"/>
            </xsd:restriction>
         </xsd:simpleType>
         <xsd:simpleType name="IP">
          <xsd:restriction base="xsd:string">
            <xsd:pattern value="(([1-9]?[0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-
5])\.){3}([1-9]?[0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])"/>
          </xsd:restriction>
         </xsd:simpleType>

</xsd:schema>
```

Appendix B
Trans-coder Schema

This appendix presents the schema for the trans-coder used in the prototype.

```
<?xml version="1.0" ?>

<!--
    Document   : transcoder.xsd
    Created on : May 18, 2004, 10:04 PM
    Organization: University of Ottawa,
    Description:
        Definition of a transcoder, it's capabilities and it's location
-->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">


    <xsd:element name="transcoder" type="TranscoderType" />
<!-- ######################### -->
<!-- Definition of Transcoder -->
<!-- ######################### -->
<xsd:complexType name="TranscoderType">
    <xsd:sequence>
       <xsd:element name="input" type="inputType" maxOccurs="unbounded"/>
       <xsd:element name="output" type="outputType" maxOccurs="unbounded"/>
       <xsd:element name="transcHardware" type="transcHardwareType" />
       <xsd:element name="proxyNode" type="proxyNodeLocation"/>
    </xsd:sequence>
    <xsd:attribute name="transcName" type="xsd:string" />
</xsd:complexType>



<!-- ############################ -->
<!-- Definition of transcHardware -->
<!-- ############################ -->
<xsd:complexType name="transcHardwareType">
    <xsd:sequence>
       <xsd:element name="processor" type="processorType" minOccurs="0" />
       <xsd:element name="memory" type="xsd:positiveInteger" />
    </xsd:sequence>
</xsd:complexType>


<!-- ######################## -->
<!-- Definition of proxyNode -->
<!-- ######################## -->
<xsd:complexType name="proxyNodeLocation">
    <xsd:attribute name="location" type="xsd:string" />
</xsd:complexType>


<!-- ########################### -->
<!-- Definition of processorType -->
<!-- ########################### -->
<xsd:complexType name="processorType">
    <xsd:sequence>
```

```
      <xsd:element name="processorName" type="xsd:string" />
      <xsd:element name="cyclesPer" type="xsd:decimal" />
    </xsd:sequence>
</xsd:complexType>


<!-- ################### -->
<!-- Definition of input -->
<!-- ################### -->
<xsd:complexType name="inputType">
    <xsd:sequence>
      <xsd:element name="inputName" type="MIMEType" />
      <xsd:element name="iproperty" type="iopropertyType" minOccurs="0"
maxOccurs="unbounded" />
    </xsd:sequence>
</xsd:complexType>


<!-- #################### -->
<!-- Definition of output -->
<!-- #################### -->
<xsd:complexType name="outputType">
    <xsd:sequence>
      <xsd:element name="outputName" type="MIMEType" />
      <xsd:element name="oproperty" type="iopropertyType" minOccurs="0"
maxOccurs="unbounded" />
      <xsd:element name="visualQuality" type="visualQualityType" minOccurs="0"
/>
      <xsd:element name="audioQuality" type="audioQualityType" minOccurs="0" />
    </xsd:sequence>
</xsd:complexType>


<!-- ######################### -->
<!-- Definition of ioproperty -->
<!-- ######################### -->
<xsd:complexType name="iopropertyType">
    <xsd:sequence>
      <xsd:element name="property" type="xsd:string" />
       <xsd:element name="value" type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>


<!-- ############################ -->
<!-- Definition of visualQuality -->
<!-- ############################ -->
<xsd:complexType name="visualQualityType">
    <xsd:sequence>
      <xsd:element name="videoFormat" type="xsd:string" minOccurs="0" />
      <xsd:element name="Pixel" type="PixelType" minOccurs="0" />
      <xsd:element name="Frame" type="FrameType" minOccurs="0" />
    </xsd:sequence>
</xsd:complexType>


<!-- ##################### -->
<!-- Definition of Format -->
<!-- ##################### -->
<xsd:complexType name="FormatType">
    <xsd:attribute name="FormatName" type="xsd:string" use="optional" />
    <xsd:attribute name="colorDomain" use="optional" default="color">
    <xsd:simpleType>
```

```
      <xsd:restriction base="xsd:NMTOKEN">
           <xsd:enumeration value="binary" />
           <xsd:enumeration value="color" />
           <xsd:enumeration value="graylevel" />
           <xsd:enumeration value="colorized" />
      </xsd:restriction>
    </xsd:simpleType>
    </xsd:attribute>
</xsd:complexType>


<!-- ################### -->
<!-- Definition of Pixel -->
<!-- ################### -->
<xsd:complexType name="PixelType">
    <xsd:attribute name="resolution" type="xsd:nonNegativeInteger"
use="optional" />
    <xsd:attribute name="aspectRatio" type="nonNegativeReal" use="optional" />
    <xsd:attribute name="bitsPer" type="xsd:nonNegativeInteger" use="optional"
/>
</xsd:complexType>


<!-- ################### -->
<!-- Definition of Frame -->
<!-- ################### -->
<xsd:complexType name="FrameType">
    <xsd:attribute name="height" type="xsd:nonNegativeInteger" use="optional"
/>
    <xsd:attribute name="width" type="xsd:nonNegativeInteger" use="optional" />
    <xsd:attribute name="aspectRatio" type="nonNegativeReal" use="optional" />
</xsd:complexType>


<!-- ########################## -->
<!-- Definition of audioQuality -->
<!-- ########################## -->
<xsd:complexType name="audioQualityType">
    <xsd:sequence>
       <xsd:element name="audioFormat" type="xsd:string" minOccurs="0" />
       <xsd:element name="audioChannels" type="audioChannelsType" minOccurs="0"
/>
       <xsd:element name="audioSample" type="audioSampleType" minOccurs="0" />
    </xsd:sequence>
</xsd:complexType>


<!-- ########################## -->
<!-- Definition of audioChannels -->
<!-- ########################## -->
<xsd:complexType name="audioChannelsType">
    <xsd:attribute name="front" type="xsd:nonNegativeInteger" use="optional" />
    <xsd:attribute name="rear" type="xsd:nonNegativeInteger" use="optional" />
    <xsd:attribute name="side" type="xsd:nonNegativeInteger" use="optional" />
    <xsd:attribute name="lfe" type="xsd:nonNegativeInteger" use="optional" />
    <xsd:attribute name="track" type="xsd:nonNegativeInteger" use="optional" />
</xsd:complexType>


<!-- ######################### -->
<!-- Definition of audioSample -->
<!-- ######################### -->
<xsd:complexType name="audioSampleType">
```

```
    <xsd:attribute name="rate" type="nonNegativeReal" use="optional" />
    <xsd:attribute name="bitsPer" type="xsd:nonNegativeInteger" use="optional"
/>
    <xsd:attribute name="Stereo" type="xsd:boolean"/>
</xsd:complexType>


<!-- ######################### -->
<!-- Definition of MIME Type -->
<!-- ######################### -->
<xsd:simpleType name="MIMEType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="text/.*" />
      <xsd:pattern value="multipart/.*" />
      <xsd:pattern value="message/.*" />
      <xsd:pattern value="application/.*" />
      <xsd:pattern value="image/.*" />
      <xsd:pattern value="audio/.*" />
      <xsd:pattern value="video/.*" />
      <xsd:pattern value="model/.*" />
    </xsd:restriction>
</xsd:simpleType>


<!-- ############################## -->
<!-- Definition of nonNegativeReal -->
<!-- ############################## -->
<xsd:simpleType name="nonNegativeReal">
    <xsd:restriction base="xsd:double">
      <xsd:minInclusive value="0" />
    </xsd:restriction>
</xsd:simpleType>


</xsd:schema>
```

Appendix C

Intermediary Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--Document   : proxyNodeDefinition.xsd
    Created on : May 10, 2004, 9:22 AM
    Organization: University of Ottawa
    Description:  Describes the information of a proxy node that contains
transcoders.
-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
       <xsd:element name="proxyNode" type="proxyNodeType" />
       <!-- ######################## -->
       <!-- Definition of proxyNode -->
       <!-- ######################## -->
       <xsd:complexType name="proxyNodeType">
              <xsd:sequence>
                   <xsd:element name="transcoderDirectory" type="xsd:string" />
                      <xsd:element name="IP" type="xsd:string"/>
                      <xsd:element name="pathPort" type="xsd:positiveInteger"/>
              <xsd:element name="neighborList" type="neighborListType"
minOccurs="0"/>
                        <xsd:element name="type" type="xsd:string"/>
              </xsd:sequence>
       </xsd:complexType>
       <!-- ########################### -->
       <!-- Definition of neighbor List -->
       <!-- ########################### -->
       <xsd:complexType name="neighborListType">
          <xsd:sequence>
              <xsd:element name="neighbor" maxOccurs="unbounded">
                 <xsd:complexType>
                   <xsd:attribute name="neighborDir" type="xsd:string"/>
                     <xsd:attribute name="bandwidth" type="xsd:positiveInteger"
/>
                   <xsd:attribute name="dataPort" type="xsd:positiveInteger" />
                   <xsd:attribute name="pathPort" type="xsd:positiveInteger" />
                   <xsd:attribute name="IP" type="xsd:string" />
                 </xsd:complexType>
              </xsd:element>
          </xsd:sequence>
       </xsd:complexType>
</xsd:schema>
```

Appendix D

Acronyms

DSRG: Distributed System Research Group
ABR: Available Bit Rate
ALS: Application Layer Signaling
APC: Automatic Path Creation Service
ARPAnet: U.S. Defense Department network
AS: Audio Service
ASA: Application-Specific Address
BGP: Border Gateway Protocol
Bluetooth SDP: Bluetooth Service Discovery Protocol
CA: Communication Agent
CC/PP: Composite Capability/Preference Profile
CMA: Call Management Agent system
CMAP: CMA Protocol
CoS: Class of Service
CPExchange: Customer Profile Exchange
CPL: Call Processing Language
CTSpec: Communication Terminal Specification
DAG: Directed Acyclic Graph
DiffServ: Differentiated Service
DNS: Domain Name Service
DS: Directory Service
DSG: Destination Set Grouping
DSRG: Distributed System Research Group
DWDM: Dense Wave Division Multiplexing
FDA: Foreign Directory Agent
FEC: Forward Equivalence Class
HD: Home Directory
HDA: Home Directory Agent
HDTV: High Definition TV
Hyper-Text Transport Protocol
IAP: ICEBERG Access Points
ICEBERG : Integrated Communications
IN: Intelligent Network
IntServ: Integrated Service
iPOP: ICEBERG Points of Presence
ISP: Internet Services Providers
ITU: International Telecommunication Union
iUID: ICEBERG Unique ID
J2SE: Java 2 platform Standard Edition

J2SE: Java 2 Standard Edition
JMF: Java Media Framework
LAN: Local Area Network (LAN),
LC: Logical Channel
LDP: Label Distribution Protocol
LDU: Logical Data Units
LER: Label Edge Router
LIB: Label Information Base
LSP: Label Switch Path
LSR: Label Switch Router
MAS: Multi Agent Systems
Mbone: Multicast Backbone
MC: Multi-point Controller
MCU: Multipoint Control Unit
MG: Media Gateway (MG)
MGC: Media Gateway Controller
MGCP: Media Gateway Control Protocol
MobInTel : Mobile Internet Telecommunication
MP: Multi-point Processor
MPA: Mobile People Architecture
MPLS: Multi-Protocol Label Switching
NRC: National Research Council of Canada
OCoV: Optimal Combination of Variants
OPE: Oblivious Polynomial Evaluation
OT_1^2 : Oblivious Transfer
P2P: Peer-to-Peer
P3P: Platform for Privacy Preferences
PA: Personal Agent
PAC: Personal Activity Coordinator
PASA: Proxy Application-Specific Address
PC: Personal Computer
PDA: Personal Digital Assistant
PKI: Public Key Infrastructure
POID: Personal Online Identifier
POTS: Plain Old Telephone Service
PSTN: Public Switched telephone Network
QoE: Quality of Experience
QoS: Quality of Service
QSNA: QoS Selection and Negotiation Agent
RAS: Registration, Admission, and Status
RDF: Resource Description Framework
RFC: Request For Comments
RLM: Receiver-driven Layered Multicast
RSA: Rivest Shamir-Adelman
RSVP: Resource reSerVation Protocol
RTCP: Real-time Control Protocol

RTSP: Real-time Streaming Protocol
SAMM: Source Adaptive Multi-Layered Multicast (SAMM)
SAP: Session Announcement Protocol
SDA: Service Discovery Agent
SDP: Session Description Protocol
SDPng: Session Description and Capability Negotiation
SPIN: Seamless Personal Information Networking
SR: Service Registry
SSRC: Synchronization SouRCe identifier
TCP: Transport Control Protocol
TOPS: Telephony Over Packet networkS
ToS: Type of Service
UA: User Agent
UAC: User Agent Client
UAProf: User Agent Profile
UAS: User Agent Server
UCA: User Context Agent
UDP: User Datagram Protocol
UMTS: Universal Mobile Telecommunications System
UPT: Universal Personal Telecommunication
URI: Uniform Resource Identifier
VHE: Virtual Home Environment
VoIP: Voice over the Internet Protocol
VPN: Virtual Private Network
VS: Video Service
W3C: World Wide Web Consortium
WML: Wireless Markup Language
WPAN: Wireless Personal Area Networking
XML: Extensible Markup Language