Personalized options on no answer conditions

-introducing HTTP based menues into IP Multimedia Telephony

Kristine Bolstad

Cand. Scient. thesis
May 2001

University of Oslo
Department of Informatics



Foreword

The work done in this paper is 2/3 of the work leading up to the Cand. Scient. degree at the University of Oslo, Institute of Informatics.

Time and effort were spent on reading, analyzing, understanding, implementing and documenting a supplementary service based in an IP Telephony system. This could not have been done without the contribution of several people sharing their knowledge and time.

I want to thank my supervisor Lill Kristiansen for giving me the opportunity to write this paper, introducing me to a wealth of ideas and possibilities. I will also like to thank Arild Nilsen for setting up the implementation environment, and Jørn Tilley Jackobsen for valuable help with the work done on database. I will also thank Jo Herstad for encouragement and help in the writing of the report. Thanks also to Kevin Kliland, Espen Skjæran, Knut Farner and all my friends for valuable comments and corrections.

I would like to thank my mother for all the help, and Marte for being so patient with her mom.

Oslo, April 2001

Kristine Bolstad

Abstract

I have in this paper described some existing networks, PSTN and GSM, and some emerging networks, GPRS and UMTS. Taking the architecture of these networks into consideration I have presented some existing services and supplementary services based on the possibilities these networks and architecture design offer. With this as a basis I have looked closely into IP Telephony standards, architecture, possibilities and limitations. I have presented IP Telephony systems, based on the H.323 standard in general, and further more presented an actual IP Telephony system implementing this, the Ericsson IP Telephony system.

Based on the design of this system I have analyzed and prototyped a supplementary service based on the upcoming H.323 standardization Annex K, which provide an optional way for controlling user directed supplementary services in an H.323 environment.

The core of the service implemented, the OnNoAnswer service, is an alternative way of responding to events when the called party is not answering an initiated call. Instead of offering a directed forwarding service, the OnNoAnswer service offers the calling party that initates the call means for an optional way of proceeding the call, that is by sending email, sms, connection to voice-mail or other devices. For the called party, the subscriber, this service offer greater flexibility, control and screening capabilities, since he is the one controlling the proceeding options being given to the caller.

Contents

1 Introduction **1**

1.1 What the paper is all about 1

1.2 IP Telephony 2

1.3 Why is IP Telephony interesting? 4

1.4 The OnNoAnswer Service 4

1.4.1 Motivation 4

1.5 The methods and means 6

1.5.1 Literature studies 6

1.5.2 Prototyping 6

1.6 Structure of this paper 8

2 Some Existing Networks **11**

2.1 PSTN/ISDN 11

2.2 GSM 14

2.2.1 GSM architecture 14

2.2.2 Call routing in GSM 16

2.3 GPRS 18

3 Some Future Networks **21**

3.1 UMTS 21

3.2 Wireless Internet Network Communication Architecture . . . 22

4 IP Telephony **27**

4.1 The H.323 standard 27

4.1.1 Protocols 28

4.1.2 Architectural issues 29

4.1.3 Basic call completion in H.323 31

4.1.4 Call Sequence in H.323 31

4.1.5 H.323 future features 33

4.2 IP Telephony and Mobility 36
4.2.1 Terminal mobility 36
4.2.2 User mobility 37
4.2.3 Service Mobility 39

5 H.323 Annex K **41**
5.1 Use of H.323 in Annex K 42
5.2 Use of HTTP in Annex K 43

6 The Hypertext Transfer Protocol **45**
6.1 Overall operation 45
6.2 HTTP Header Fields 46
6.3 Requests 48
6.3.1 The GET method 49
6.3.2 The POST method 49
6.4 CGI and servlets 51
6.5 The Hypertext Transfer Protocol - Next generation 54

7 The Ericsson IP Telephony System **55**
7.1 Architectural issues 55
7.2 IP Telephony system overview 57
7.2.1 Access Agent 57
7.2.2 Service Node 58
7.2.3 Sitekeeper 58
7.2.4 Management 59
7.3 The half call model 59
7.4 Traffic handling 59
7.4.1 User authentication 61
7.4.2 Terminal registration 63

8 Designing a Supplementary Service **65**
8.1 The Service Agent 66
8.1.1 The Service Creation Environment 66
8.1.2 The Service Execution Environment 66
8.1.3 Call Control API 68
8.1.4 Session related interaction 68
8.1.5 Connection related interaction 69
8.1.6 Supplementary Services 69
8.2 The Subscriber and Service Data Repository 69
8.3 Technical outline 71

9 Some Existing Services **73**
9.1 Traditionally supplementary services 73
9.1.1 PSTN/ISDN supplementary services 73
9.1.2 GSM supplementary services 74
9.2 IN 74
9.2.1 IN architecturefor PSTN/ISDN 75
9.2.2 IN architecture for GSM 77
9.2.3 IN services in PSTN/ISDN 77
9.2.4 IN services in GSM 79
9.3 IP Telephony services 79
9.3.1 The Personal Agent services 80
9.4 Convergence and the future 82
9.4.1 iPulse 82
9.5 Migration 84
9.6 Some service prototyping using OSA 84

10 Implementing the Service **85**
10.1 The Service Architecture 86
10.2 Use Cases 88
10.3 Resolving the execution order 89
10.4 Retrieving Information 92
10.5 Posting of the service 93
10.6 On proceeding the call 96
10.7 Problems during the implementation phase 97
10.8 Limitations in the implementation of the OnNoAnswer Service 98

11 Usability Testing **101**
11.1 Designing the interface 102

12 Conclusion **105**
12.1 The functionality 106
12.2 Technological aspects 106
12.3 Further work 108

Bibliografi **111**

Appendix **115**

A HTML-code **I**

A.1 HTML-page presented to the subscriber I

A.2 HTML-page presented to the caller if not the subscriber . . . III

B Database **V**

B.1 Service.xsql V

B.2 Service.dtd VI

B.3 Niam Diagrams of the Database solution VII

C OnNoAnswer **IX**

C.1 OnNoAnswer plugin IX

C.2 Test Servlet XIV

List of Figures

2.1 Overview of an typical Public Switched Telephone Network (PSTN) from [GLASS] 12
2.2 Setup of an Integrated Switched Digital Network (ISDN) connection from [GLASS] 13
2.3 Overview of GSM architectural issues from [TAHN] 15
2.4 GPRS and UMTS from [TAHN] 20

3.1 Overview of an OSA UMTS architectural issues [DIP] 22
3.2 UMTS PS and IM subsystem from [3GPP] 23
3.3 IM subsystem from [3GPP] 23
3.4 Overview of existing and new IP architecture [IECW] 24

4.1 Sequence of VoIP connection [IECIP] 29
4.2 Overview of architectural issues for H.323 standard from [ITU2000] 30
4.3 Overview of architectural terminal issues for H.323 standard from [Dat98] 31
4.4 Overview of a basic call completion in an hypothetical H.323 network [KOR] 32
4.5 H.323 call sequence [IECIP] 33

5.1 Annex K system overview for HTTP based service control from [ITUTD-13] 42
5.2 Call related service control, gatekeeper routed call based on [ITUTD-13] 43

6.1 HTTP client/server overview from [BASHT] 46
6.2 HttpServlet interaction between client and server from [IETF97] 53

7.1 Overview of architectural issues from [IPTER] 56
7.2 Overview of the Ericsson IP Telephony system from [KLI] . . 57

7.3 Overview of the half call modell from [IPTER] 60
7.4 Overview of user Authentication from [IPTER] 62
7.5 Overview of terminal and user registration from [IPTER] . . . 64

8.1 Overview of the Service Agent from [NILS$^+$1999] 67
8.2 Overview of the Subscriber and Service Data Repository from [NILS$^+$2000] 70

9.1 Overview of IN architectural issues from [SKO] 75
9.2 Overview of the CAMEL architecture from [SKO] 76
9.3 Overview of the Personal Agent service from [IPTER] 80
9.4 iPulse display with menus and buttons from [IWP] 82
9.5 Overview of service architectural issues from [TAHN] 83
9.6 Call flow using OSA [PRO] 84

10.1 Class digram for the OnNoAnswer service 87
10.2 Central use-cases for the actors 88
10.3 Sequence digram for the initiating of a call to the On-NoAnswer service 89
10.4 Web page presented the calling party 94
10.5 Illustration of the HTML-page presented to the subscriber . . 95
10.6 Sequence digram for the proceeding of the call 96

Chapter 1

Introduction

I will in this paper describe different architectures and solutions for supplementary services in an IP Telephony context in special, and some selected architectures for supplementary services in traditional telephony in general. I will design and analyze a supplementary service based in an IP Telephony environment.

Based on this I will conclude with some thesis about how different architectural approaches, application design principles and user preferences are linked together, making supplementary value added services actually value adding in respect of making the user of the service in charge of his own availability.

The work done and described in this paper is for most parts of a general character.

1.1 What the paper is all about

Internet has become more common and, the use and possibility therein, opens up for new services and possibilities.

The motivation for the work described in this paper is the introduction and discussion of Annex K, a supplement to the H.323 standard for IP Telephony. This draft describes a way for employing new services in an IP Telephony system by means of the Hypertext Transfer Protocol.

By looking into existing networks and future technology, the concept of a new services based in an IP Telephony network system has been analyzed and prototyped. The service in question in this paper is a supplementary service related to call handling. That is, the service is designed to give the users more flexibility and control over traditional call processing.

Supplementary service in this context is offered through subscriptions

and maintained by an operator. Variants of such services might also be offered by a third party having a relationship with an operator, by [IECW] defined as value-added services. In the rest of this paper supplementary services is used, even though the service is “untraditional” and not always standardized.

1.2 IP Telephony

IP telephony is the technology that allows transforming any IP network into a media for real-time transfer of long-distance and international voice calls and faxes.

IP implements the network layer of the TCP/IP protocol, which contains a network address and is used to route a message to a different network or subnetwork. IP accepts “packets” from the transport protocol (TCP or UDP), adds its own header to it and delivers a “datagram” to the data link protocol. It may also break the packets into fragment to support the maximum transmission unit (MCU) of the network.

The ultimate objective of IP Telephony is reliable, high-quality voice service, the kind that users expect from the PSTN. At the moment, however, that level of reliability and sound quality is not available on the Internet, primarily because of bandwidth limitations that lead to packet loss. In voice communications, packet loss shows up in the form of gaps or periods of silence in the conversation, leading to a clipped-speech effect that is unsatisfactory for most users and unacceptable in business communications [IECIP].

In addition, because the Internet is a packet-switched network, the individual packets of each voice signal travel over separate network paths for in the proper sequence at their ultimate destination. While this makes for a more efficient use of network resources than the circuit-switched PSTN, which routes a call over a single path, it also increases the chances for packet loss [IECIP].

Quality of Service (QoS) refers to the capability of a network to provide better service, such as routing algorithms to find paths through the network to minimize packet loss. Primary goals of QoS include dedicated bandwidth, controlled jitter and latency, required by some real-time and interactive traffic, and improved loss characteristics [CIS].There are several consortiums, forums and groups working on enhancing the QoS over IP networks, making IP networks reliable for sending voice as data.
necessary

Although QoS raises important issues to be solved, IP Telephony enables a wealth of new service possibilities [Ros99]. Traditional telephony services, such as call forwarding, transfer, and 800 number services, can be enhanced by adding integration with e-mail, web, presence, instant

messaging and directory services. Users can have redirected to web-pages, use streaming media tools to record voice-mail, or use instant messages in place of call waiting notifications. IP Telephony can offer improved speech quality through advanced speech and audio , and communication can encompass not just voice, but video, shared applications, and even virtual reality. Much more powerful user interfaces can allow these services to be made easily accessible. Gateways to the Public Switched Telephone Network (PSTN) can allow these services to extend to traditional fixed phones, cellular phones and pagers.

There are two preconditions to an IP Telephony system, packet-based networks and packet based telephony standards. IP Telephony has it offspring in existing PSTN solutions and new development, and today there are three main protocols currently used for signaling IP Telephony services. These are H.323, Media Gateway Control Protocol (MGCP) and Session Initiation protocol (SIP).

There is a strong connection between programming IP Telephony services and the protocols used to deliver these services. A number of different protocols are needed for delivery of IP telephony services. These includes transport protocols such as Real Time Transport Protocol (RTP), used to carry voice over IP network, QoS protocols, such as Resource Reservation protocol (RSVP) and Differentiated Services, used to provide low delay and loss of voice transport, and directory access protocols.

Additional media types, like video and interactive chat, can be embedded as well. One of the challenges in providing these services is how to effectively program them. Programming these services requires decisions regarding where the code executes, how it interfaces with the protocols that deliver the services. More importantly are the signaling protocols. Signaling protocols are used to setup and tear-down the call, carry information required for the call to progress, locate the user to be called, negotiate capabilities, and invoke services like hold, mute, and transfer. These are the protocols that ultimately provide services [Ros99].

H.323 was developed in the International Telecommunications Union (ITU). It was originally conceived for multimedia conferencing on a LAN, but has since been extended to cover Internet Telephony. It provides call control, conference function, call management, capabilities negotiation, and supplementary services. MGCP is a control protocol. SIP was developed in the Internet Engineering Task Force (IETF), and is a protocol engineered for distributed call control and capabilities negotiation.

The work done is this paper is based on an IP Telephony system based on the H.323 protocol.

1.3 Why is IP Telephony interesting?

The autumn of 2001 Sweden is starting up test-projects with IP Telephony. After this test-period the Swedish Teletilsyn (PTS) is considering connecting tele and Internet together permanently. IP telephony has been experimented with and developed since 1970, but it has not yet been a complete success. Still, there are several aspects of IP Telephony that makes the solution so interesting. Initial it was the idea of cutting costs that was most appealing, now it seems like the wealth of possible new services are the most appealing.

Some claims that price and cost will vary, and that this is not an argument for IP Telephony success. Traditionally telecommunication will adapt to the competition and lower their prices. On the other hand; Computer Telephony Integration (CTI) will open up for new services, connection traditionally telephony, well known Internet services and data services into a conglomerate of new services and possibilities. [UMTSFORUM] claims that

> VoIP will become an important enabler of convergence for services in fixed and mobile networks.

IP Telephony is in most countries still not regulated in the same way as traditional telecommunication. The regulatory implications of VoIP are still not entirely clear. The Internet community believes that regulation within the Internet environment is not required. The European Commission has already announced that there is no need to regulate VoIP. Opinions on this issue differ in a number of the European Union member states [UMTSFORUM].

1.4 The OnNoAnswer Service

Based on Annex K, which describes an HTTP based service control transport channel in an H.323 environment, several new services could be embedded. The service in question in this paper has its offspring in the assumption that subscribers would like to control their own availability and also differentiate between the availability for different groups of callers.

1.4.1 Motivation

When a caller A initiates a call to called part B, B may not want to answer the call. As it works today with GSM, the called party B can identify the caller by means of number display. He can then choose to refuse the call or answer it. With ISDN the normal procedure to increase availability is *Forward Call Unconditional*, that is, all calls are routed to where the user actually is, this visiting his parents or at a friends house. *Forward to voicemail* could be activated, meaning that all calls would be forwarded

to the subscribers voice-mail.The idea behind the OnNoAnswer service is that the user should be in charge of his own availability, both in terms which calls he wants to answer, and how he wants to answer the call.

In ISDN and GSM, when the called party is not answering, the calling party would get the alerting signaling, after which he could be redirected to for example voice-mail. If the subscriber is subscribing to a forwarding service, or the terminal has forwarding logic, the call could be forwarded to another terminal. If the subscriber is not answering there either, eventually the caller initiating the call would get a voice-mail, nothing or the initiator of the call would hang up before getting this far in the process of forwarding. This terminal-centric approach connects the terminal and the services together.

In IP Telephony the user can register at several different terminals at the same time. By implementing the service logic in the network, the user can access the services from any terminal that he is registered at, getting access to whatever services he is subscribing to. This network-centric approach connects the user to the service, and not the terminal. One important aspect of the network-centric approach is that the service is located in a service node with the Service Provider.

There are two important aspects of the OnNoAnswer service:

- The control over own availability.

By giving the subscriber control over his own availability, the OnNoAnswer service offers the subscriber means by which he can give different calling parties options for how to further process the call. The calling party will be presented an HTML-page presenting a block of information saying why the call was not answered, and options for how to proceed the call. Both the information and options for proceeding the call will be determined based on identification of the calling party. The subscriber will still be in control over which options for continuing the call that is offered the caller, but the caller would be presented with more options than voice-mail or just ending the call.

Depending on capacity of the end-point from which the call is initiated the calling party could be presented options like leave an email, go to my web-page, forward to my colleague or please leave a message.

- The connection between services and user.

As users of telecommunication services are getting more mobile, the need for accessing the same set of services with the latest configurations increases. If services are implemented to reside in a terminal, the terminal is then the owner of the service. By implementing most parts of the OnNoAnswer service logic in the network the user can access the service from any terminal, as long as the terminal holds the network and media capability .

For the OnNoAnswer service the logic telling the gatekeeper that the call should be routed to the terminal or not, could be in the terminal. The call handling, and the logic for database handling will reside in the network.

Another feature introduced by Annex K is that the communication channel could be used outside an actual call. This opens for downloading applications from the network. If the terminal holds enough capacity it will then be possible to hold the application locally on the terminal, instead of generating traffic on the network.

1.5 The methods and means

1.5.1 Literature studies

The first approach to the work done described in his paper was reading literature about the telecommunication science field, since I had very little experience in this area. Through studies of telecommunication, and telecommunication services, I got a grasp of the field and the different aspects of designing a service to run in an IP Telephony environment.

Based on this, I went on reading about different networks and standards, different methods when designing a service, and usability aspects. The literature I have read have been documentation presented to me during different courses and presentations, introduced to me by my supervisors and sought for on the Internet and the Ericsson Intranet, and in libraries.

These literature studies forms the basic of the understanding, and the work done in this paper.

1.5.2 Prototyping

Since the implementation of H.323 protocol in the Ericsson IP Telephony system, in which the implementation took place, fully support some of the more important aspects of the OnNoAnswer service. It became important to prototype some of the functionality to see if the OnNoAnswer service was a feasible service, both technological and from a users point of view. For actually developing and deploy such a service in an IP Telephony system based on the H.323 protocol, changes had to be made to the protocols call set-up messages handling, and the standards had to be implemented throughout the whole system.

The solution was to implement a prototype, or more, a kind of solution that could illustrate the functionality desired. Prototyping is a central tool in a development process when the area of the problem is unfamiliar,

both in regards to technology and also in regards to user concepts [**?**].

Prototyping is an activity and a method in evolutionary system development and is recommended when there is some form on uncertainty regarding technology, users and similar factor [MAT]. Prototyping involves producing early versions of the future application. This version of the future application opens for discussion between different teams and groups involved in the design process, and also other phases in an system development project [**?**]. The hope is that the different parts involved will exchange information and attitudes towards the prototype, and by means of the discussion that will arise the prototype will evolve further in one of many possible direction, hopeful a value-adding one [SOM].

The objective of the prototype should be made explicit from the start of the project. If not, designers and end-users may misunderstand the function of the prototype. Consequently, they may not get the benefits that they expected from the prototype development. The next stage in working with a prototype is to decide what to put into and, perhaps more importantly, what to leave out. Normal practice in prototyping development is to relax non-functional requirements such as response time and memory utilization. Error handling and management may be ignored. Standards of reliability and program quality may be reduced. The final stage of the process is prototype evaluation.

Prototyping is distinguished in several forms of prototyping; horizontal by developing several layers of the application, vertical prototyping means that all layers in an application is developed for a fraction, or minor part, of the application, exploring prototyping is often deployed when there are some uncertainty about the problem area, experimenting prototyping focuses on the technical implementation, while evolutionary prototyping is a process for making a system suitable for an organization in change.

According to [SOM] evolutionary prototyping is one, if not the only, way to develop application and systems where it is difficult, or impossible, to establish a detailed specification. The key to success for this approach is to use techniques, which allows for rapid development.

The approach for the prototype of the OnNoAnswer service is for most evolutionary and experimental. Evolutionary since the application specification was not clear. Experimentally; because I was interested in both the user interaction aspect of the OnNoAnswer service, and the technical solution. The design and implementation of the OnNoAnswer service, is one of many possible aspect of the functionality the introduction of Annex K offers. By implementing the OnNoAnswer service as a prototype the functionality of the one aspect could be demonstrated for users, asking for input regarding other similar functionality.

Furthermore, the prototype could reveal technical obstacles and aspect about such a message interaction call control service, how this would work technically. Since the Ericsson IP Telephony system has not yet im-

plemented the functionality which could handle such call back interaction, the prototype, and the evaluation of the prototype would give some idea if an implementation to support such a service would be worth while.

The first phase of the protoype is described in chapter 10, while the evaluation of the prototype is described in the chapter about further work, since due to restricted time, there were no time to conduct this work.

1.6 Structure of this paper

I will shortly describe the structure of this paper. Below are the contect of the various chapter further described. I have not added an acronymlist since the acronyms used are described text.

Chapter 1 - Introduction

This chapter gives an introduction to the field IP telephony and services in telecommunication. I give an introduction the the issue in focus in this paper.

Chapter 2 - Some Existing Networks

Here I present som existing networks, PSTN, ISDN, GSM and GPRS.

Chapter 3 - Some Future Networks

This chapter presents UMTS, and Open Service Architecture.

Chapter 4 - IP Telephony

Here i present the H.323 standard, protocols, architecture, call handling and future features.

Chapter 5 - Annex K

This chapter gives an introduction and overview of the H.323/annex K that is the basic for the work done in this thesis.

Chapter 6 - The Hypertext Transfer Protocol

I have also chosen to include a chapter on HTTP and use og HTTP and Servlets.

Chapter 7 - The Ericsson IP Telephony System

The Ericsson Ip telephony system is the base for the work done with implementing the OnNoAnswer service. This chapter outlines some major issues in this system.

Chapter 8 - Designing a Supplementary Service

Here I describe the theoretical foundation under which the service were implemented.

Chapter 9 - Some Existing Services

To better understand the features of the OnNoAnswer service I have illustrated a few traditional services offered to end users.

Chapter 10 - Implementing the Service

This chapter describes the work done on implementing the OnNoAnswer service.

Chapter 11 - Usability Testing

I have also included a chapter on usability. here I point to some important usability issues to consider when designing a interactive service for end-users.

Chapter 12 - Conclusion

This last chapter sums up the the theories presented and the work done in this thesis. I point some of the most central findings, and concludes with some directions for further work.

Chapter 2

Some Existing Networks

2.1 PSTN/ISDN

PSTN can be explained as a domestic telecommunications network usually accessed by telephones, key telephone systems, private branch exchange trunks, and data arrangements. Completion of the circuit between the call originator and call receiver in a PSTN requires network signaling in the form of dial pulses or multifrequency tones.

To make a call through a PSTN network the telephone of the calling party sends a signal over the copper wire to the local exchange. The local exchange responds by providing a dial tone to the telephone. The telephone then sends signaling information by the local exchange containing the dialed number, to identify the party called. Signaling is done via SS7. Signaling System 7 (SS7) is an architecture for performing out-of-band signaling in support of the call-establishment, billing, routing, and information-exchange functions of the public switched telephone network (PSTN). It identifies functions to be performed by a signaling-system network and a protocol to enable their performance [IECS].

The local exchange recognize the dialed number and establishes a connection which, if the called party is connected to the same local exchange, will be a copper wire directly from the local exchange to the called party, or, if to a more distant destination, will be through a trunk transmission medium to another local exchange, perhaps via one or more trunk exchanges. there could be several possible routes, or series of connections, which can effect a connection between the calling party and the party called [GLASS]. Figure **??** on page ?? gives an overview of call flow in a PSTN Network.

As there is no real ISDN network itself a discussion about ISDN means basically speaking about the user/network interface [TIR]. To be able to use the services provided by ISDN over network boundaries the signaling system SS7 had to be extended. This was done by introducing the ISDN user part (ISUP). It refers to signaling activities of the network triggered

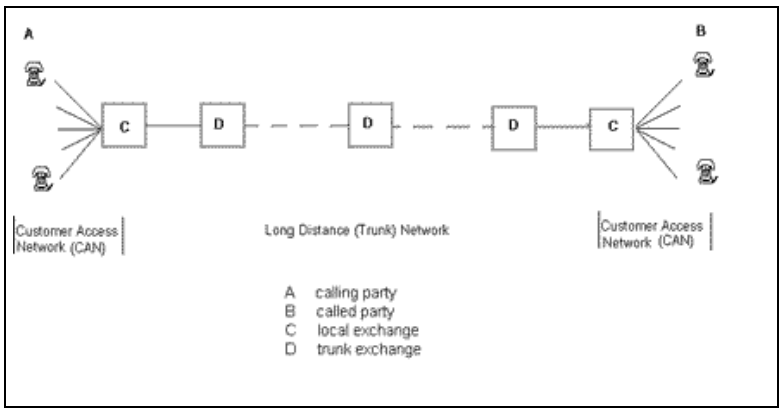


Figure 2.1: Overview of an typical Public Switched Telephone Network (PSTN) from [GLASS]

by the user and is defined in ITU-T Recommendations Q.761[35]-Q.764[36].

The standards describing the user/network interface are defined in the Q.92x, Q.93x, Q.94x and Q.95x series of the ITU recommendations. The standards in the Q.92x series mainly deal with the data link layer and its protocol whereas the recommendations in the Q.93x take care about the protocols of the network layer.

The standards described above cover the basic call control of ISDN connections. But supplementary services form an integrated part of an ISDN network. These services in connection with the ISDN D channel protocol are treated in Q.932. Q.932 is an integral part of Q.931. It adds messages and functionality to Q.931 in order to provide a means to access and control supplementary services over the user/network interface.

The Q.95x series of the ITU recommendations specify the actual supplementary services. In these standards each service is defined and described. Operation and coding requirements are also given. Moreover, interactions with other supplementary services and networks are covered. Supplementary services are described in chapter 9, and are mostly executed in local exchanges, see the C-blocks in figure 2.1

When the ISDN was designed its developers wanted it to be as flexible as possible. This meant the subscriber side should be free to configure his system in accordance to his needs and without restrictions. If the subscriber decides to connect another terminal to the system or disconnect a telephone he should be able to. In order to provide this flexibility the system was designed in a way that did not require any knowledge about the subscribers equipment in the local exchange. The only requirement is that the equipment connected to the network understands the protocols used in ISDN.

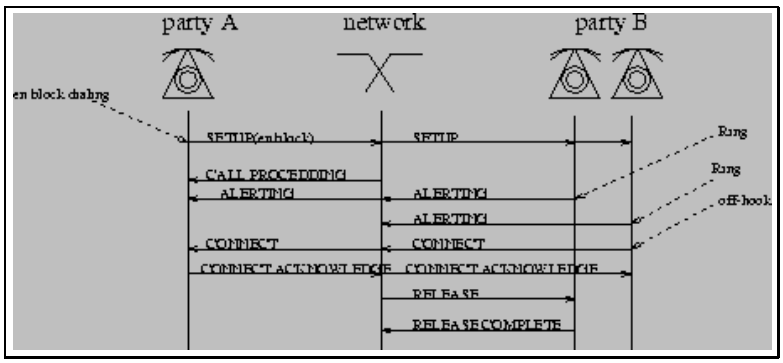


Figure 2.2: Setup of an Integrated Switched Digital Network (ISDN) connection from [GLASS]

During the call setup phase the switch sends a SETUP message on the D channel to the subscriber.The body of this message contains information on a variety of issues that is necessary to complete the setup in a way that fulfills the requirements of the intended connection. It would not make any sense to connect a telephone with a data terminal or a fax machine with a telephone.

The supplementary services is an part of ISUP. Following figure 2.1 on the facing page services as Calling Line Identity Presentation (CLIP) and Calling Line Identity Restriction (CLIR) determined by local exchanges (left C-block) and transported via trunk exchanges (D-blocks) to the calling party's local exchange (right C-block). The same goes for forwarding services further illustrated in chapter 9.

To perform any actions on the network layer level, a valid layer 2 connection has to be established between the switch and every piece of user equipment. After physically connecting a new device to the network it will start to establish such a connection to the network. Again: the user equipment has to initiate a layer 2 connection to the switch because the switch does not have any information about the configuration on the user side.

What happens is that the setup message is received by every piece of ISDN equipment connected to the ISDN subscriber line. Each terminal, phone etc. decides if it is capable of accepting the call or not. Only if the equipment is able to provide the service specified in the bearer capability info element will it send the ALERTING message back to the switch. Only the first ALERTING message is of any importance for the switch. It is clear that on the user side is equipment available to accept the desired connection when the first ALERTING message has been received. Further ALERTING messages sent by other equipment as an answer on the same SETUP message on the same line will be ignored by the switch, se figure 2.2.

2.2 GSM

During the early 1980s, analog cellular telephone system were experiencing rapid growth in Europe. Each country developed its own system, which was incompatible with everyone else's in equipment and operation. This situation was undesirable, and in 1982 the Conference of European Posts and Telegraphs (CEPT) formed a study group called the Groupe Spècial Mobile (GSM) to study and develop a pan-European public land mobile system. The acronym later changes to stand for Global System for Mobile communications (GSM).

The over 8000 pages of GSM recommendations try to allow flexibility and competitive innovation among suppliers, but provide enough standardization to guarantee proper inter-working between the components of the system. This is achieved by providing functional and interface description for each of the functional entities defined in the system [TAHN]

GSM had to meet certain criteria. Some of these where low terminal and service cost, support for international roaming, support for range of new services and facilities and ISDN compatibility in terms of the service offered and the control signal used. However, radio transmission limitations, in terms of bandwidth and cost, do not allow the standard ISDN B-channel bit rate of 64 kbps to be practically achieved [**?**], but instead a rate of 9.6 kbps, a fact that puts some restrictions on the service integration between GSM and ISDN [WAL]

A unique feature in the GSM is the Short Message Service (SMS). The Short Message Service (SMS) is the ability to send and receive text messages to and from mobile telephones. The text can comprise of words or numbers or an alphanumeric combination. SMS was created as part of the GSM Phase 1 standard. The first short message is believed to have been sent in December 1992 from a Personal Computer (PC) to a mobile phone on the Vodafone GSM network in the UK [GSMWORLD]. SMS is a bi-directional service for short alphanumeric, up to 160 bytes, messages. The messages are transported in a store-and-forward fashion. For point-to-point SMS, a message can be sent to another subscriber to the service, and an acknowledgment of receipt is provided to the sender. SMS can also be used in a cell-broadcast mode, for sending messages such as traffic updates or news updates. This service is provided by an external node in the GSM network system [WAL].

2.2.1 GSM architecture

An overview of the GSM architecture is given in figure 2.3 on the facing page. The GSM network can be divided into three:

- The Mobile Station (MS) that is carried by the subscriber consists of

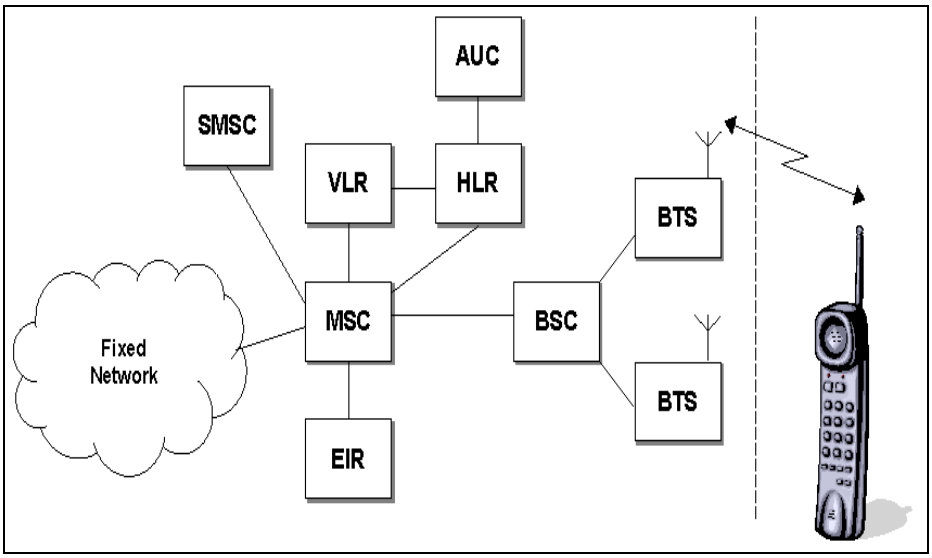


Figure 2.3: Overview of GSM architectural issues from [TAHN]

the actual terminal and a smart card called the Subscriber Identity Module (SIM). The SIM provides personal mobility, so that the user can have access to subscribed services irrespective of a specific terminal. By inserting the SIM card into another GSM terminal, the user is able to receive all calls at that terminal, make calls from that terminal, and receive other subscribed services.

The SIM card contains the International Mobile Subscriber Identity (IMSI) used to identify the subscriber to the system, a secret key for authentication, and other information. The mobile equipment is uniquely identified by the International Mobile Equipment Identity (IMEI). The IMEI and the IMSI are independent, thereby allowing personal mobility.

- The BaseStation Subsystem (BSS) is composed of two parts, the Base Transceiver Station (BTS) that houses the radio transceivers that define a cell and handles radio-link protocols with the Mobile Station (MS), and the Base Station Controller (BSC) that manages the radio recourses for one or more BTSs. The BSC is the connection between the Mobile Station (MS) and the Mobile service Switching Center (MSC). These communicates across the standardized Abis interface, allowing operations between components made by different suppliers.

- The Network Subsystem, of which the main part is the Mobile service Switching Center (MSC). The MSC performs the switching of calls between the mobile users, and between mobile and fixed network users. It acts like a normal switching node of the PSTN or ISDN, and additionally provides all the functionality needed to handle a mo-

bile subscriber, such as registration, authentication, location update, handovers, and call routing to a roaming subscriber. these services are provided in conjunction with several functional entities, which together form the Network Subsystem. The MSC provides the connection to the fixed networks such as PSTN and ISDN.

Further more the Home Location register (HLR) and Visitor Location Register (VLR) together with the MSC, provide the call-routing and roaming capabilities of GSM. The HLR contains all the administrative information of each subscriber registered in the corresponding GSM network, along with the current location of the mobile. The location of the mobile is typically in form of the signaling address of the VLR associated with the Mobile Station (MS). There is logically one HLR per GSM network, although is may be implemented as a distributed database.

The Visitor Location Register (VLR) contains selected information from the HLR necessary for call control and provisioning of the subscribed services, for each mobile currently located in the geographical area controlled by the VLR. Although each functional entity can be implemented as an independent unit, all manufactures of switching equipment to date implement the VLR together with the MSC, so that the geographical area controlled by the MSC corresponds to that controlled by the VLR, thus simplifying the signaling required. The MSC thus holds no information about particular Mobile Stations (MS) since this information is stored in the location registers.

The Equipment Identity Register (EIR) is a database that contains a list of all valid mobile equipment ion the network, where each Mobile Station (MS) is identified by its IMEI.

The Authentication Center (AuC) is a protected database that stores a copy of the secret key stored in each subscribers SIM card, which is used for authentication and encryption over the radio channel.

Supplementary services in GSM are primarily voice services and are usually the first to be offered by a network operator after their launch. Examples of supplementary services in a GSM network is barring of calls, several call forwarding options and voice-mail [GSMWORLD].

Receiving and sending data is the essential building block leading to widespread mobile Internet access and mobile data transfer. The GSM Short Message Service (SMS) is the ability to send and receive text messages to and from mobile telephones

2.2.2 Call routing in GSM

I will in this section describe an incoming call routing scenario from an PSTN terminal to a cellular terminal. Specific functionality must be invoked in order to route an incoming call to a cellular terminal since

the cellular network support a single calling number irrespective of the whereabouts of the called mobile.

- The fixes public network recognizes that the called number is a mobile number and routes the call to a mobile switch.
- The mobile switch interrogates that location database in order to retrieve routing information
- A roaming number enabling routing according to the current position of the called number is returned.
- The call is routed to the correct mobile switch, often through the fixed network.
- The mobile switch controlling the current area is searching for the called mobile, and the call is established is a cell depending on the actual position of the mobile.

A critical point in the routing performance context is at which point the call is recognized to be a special call with a need for interrogation og routing information.

In the public land networks of today the only generally available way to recognize a call being of this category is to analyze the called number and routing it to a mobile switch. Unfortunately, this will often lead to a so-called tromboning especially in the case of national roaming users.

[WAL] and Lill Kristiansen has illustrated this by the following scenario:

Consider the example of a Telenor Mobile GSM user roaming in cellular network in France. A person in France who +47 900 xxxxx would get a circuit to an international exchange in Norway, since the French network has no knowledge og the numbering structure behind the Norwegian country code. According to [TAHN] there are also countries which do not separate mobile numbering series from other numbering series. Then the call will be routed to a Telenor Mobile service Switching center (MSC) and back the same way to an MSC in France servicing the area in which the mobile resides, that is the called party is also located in France along with the calling party.

Thus, the call occupies two international links while a short signaling interrogation would have been all that was needed in addition to the local call. This inefficiency will be overcome when mobility in included also in the fixed networks. Look-ahead procedures in different forms will be implemented, according to [WAL]

A step in this direction is taken by the development of the Intelligent

Network (IN) concepts. The IN architecture for fixed network and GSM will be further described later in this paper.

2.3 GPRS

The general Packet Radio Service (GPRS) is a new non-voice value added service that allows information to be sent and received across a mobile telephone network. It supplements today's Circuit Switched Data and Short Message Service [GSMWORLD].

GPRS is based on best-effort, and GPRS involves overlaying a packet based air interface on the existing circuit switched GSM network. This gives the user an option to use a packet-based data service. To supplement a circuit switched network architecture with packet switching is quite a major upgrade. However the GPRS standard is delivered in a very elegant manner- with network operators needing only to add a couple of new infrastructure nodes and making a software upgrade to some existing network elements.

With GPRS, the information is split into separate but related packets before being transmitted and reassembled at the receiving end. According to [BUCK]:

> Packet switching is similar to a jigsaw puzzle- the image that the puzzle represents is divided into pieces at the manufacturing factory and put into a plastic bag. During transportation of the now boxed jigsaw from the factory to the end user, the pieces get jumbled up. When the recipient empties the bag with all the pieces, they are reassembled to form the original image. All the pieces are all related and fit together, but the way they are transported and assembled varies.

The Internet itself is another example of a packet data network.

Packet switching in GPRS means that GPRS radio resources are used only when users are actually sending or receiving data. Rather than dedicating a radio channel to a mobile data user for a fixed period of time, the available radio resource can be concurrently shared between several users. This efficient use of scarce radio resources means that large numbers of GPRS users can potentially share the same bandwidth and be served from a single cell. The actual number of users supported depends on the application being used and how much data is being transferred. Because of the spectrum efficiency of GPRS, there is less need to build in idle capacity that is only used in peak hours. GPRS therefore lets network operators maximize the use of their network resources in a dynamic and flexible way, along with user access to resources and revenues.

For the first time, GPRS fully enables Mobile Internet functionality

by allowing interworking between the existing Internet and the new GPRS network. Any service that is used over the fixed Internet today- File Transfer Protocol (FTP), web browsing, chat, email, telnet- will be as available over the mobile network because of GPRS. In fact, many network operators are considering the opportunity to use GPRS to help become wireless Internet Service Providers in their own right.

Whatever the application, the Internet will become the primary communications interface. Previously, application developers wrote proprietary applications that worked with proprietary host terminals and often proprietary rugged terminal operating systems. For example, instead of corporate applications such as service engineering using platform and software specific interfaces, the mobile workers such as service engineers will access an intranet page using their GPRS capable terminal and fill in an electronic form. People increasingly use a web browser to access publicly available data on the Internet itself, the extranet for access to the data of business partners and other external collaborators and the intranet to access internal employee information. As such, all work will be carried out through the web interface.

Often, by designing applications to minimize the effects of the limitations of existing mobile services- such as the length of a short message or the speed of a Circuit Switched Data call- existing non-voice mobile services can be successfully used for mobile working. However, many non-voice applications are graphics intensive and the new faster data services will allow BETTER VERSIONS of today's existing non-voice applications. For example, instead of occasional information messages with SMS, information services via GPRS or UMTS will be more akin to the "push" Internet channels we see on Active PC today. Instead of the slow transmission of small video images, real-time broadcast quality images will be transmittable. Instead of using SMS to notify Internet users of new email, the whole email will be sent, and full-blown Internet access will be possible. The same applications will be more immediate and convenient for users.

The use of SMS has prepared customers for non-voice applications using GPRS and other non-voice services and most of the applications envisaged for GPRS already exist in some form today. It is therefore an important question to consider what the preferred bearer for each application will be- GPRS, Circuit Switched Data or SMS.

Enabling GPRS on a GSM network requires the addition of two core modules, the Gateway GPRS Service Node (GGSN) and the Serving GPRS Service Node (SGSN). As the word Gateway in its name suggests, the GGSN acts as a gateway between the GPRS network and Public Data Networks such as IP and X.25. GGSNs also connect to other GPRS networks to facilitate GPRS roaming. The Serving GPRS Support Node (SGSN) provides packet routing to and from the SGSN service area for all users in that service area. Looking aside the part in figure 2.4 on the next page that handles UMTS, the figure illustrates a GPRS network.

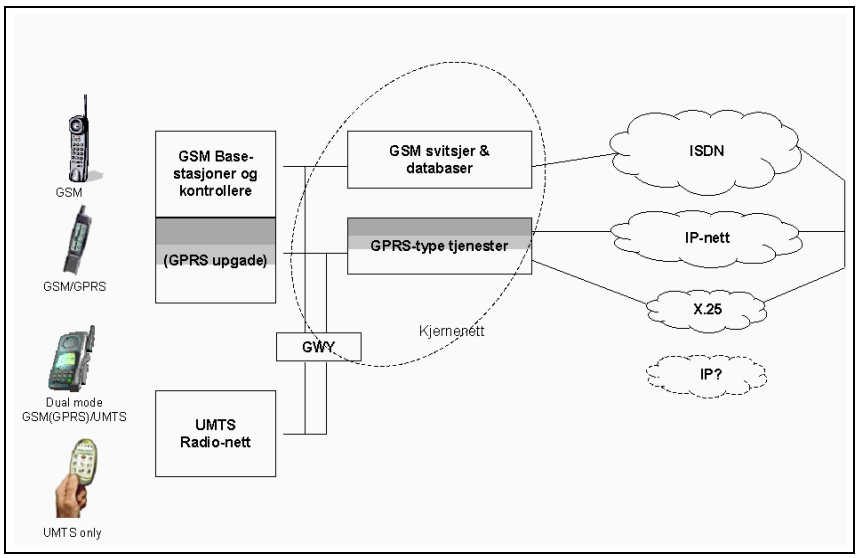


Figure 2.4: GPRS and UMTS from [TAHN]

In addition to adding multiple GPRS nodes and a GPRS backbone, some other technical changes that need to be added to a GSM network to implement a GPRS service. These include the addition of Packet Control Units; often hosted in the Base Station Subsystems, mobility management to locate the GPRS Mobile Station, a new air interface for packet traffic, new security features such as ciphering and new GPRS specific signaling.

GPRS deployed today offer best effort services well suited for SMS, e-mail, Internet etc. Later GPRS will offer IP based streaming with some QoS mechanisms. Later GPRS will evolve into full QoS mechanisms and will be used in connection with UMTS.

Chapter 3

Some Future Networks

3.1 UMTS

UMTS is an acronym for the Universal Mobile Telecommunication System. UMTS is part of the International Telecommunications Union's IMT-2000 vision of a global family of third generation (3G) mobile communication system. It is said that UMTS will play a key role in creating future mass market for high-quality wireless multimedia communication.

Further more, UMTS is said to deliver pictures, graphics, video communications and other wide-band information as well as voice and data, direct to people who can be on the move. UMTS will build on and extend the capability of today's mobile technologies (like digital cellular and cordless) by providing increased capacity, data capability and a far greater range of services using an innovative radio access scheme and an enhanced, evolving core network.

The telecommunication market has recognized two clear trends [UMTSFORUM]. That is; users have an increasing need to gain access to information in an efficient manner as possible, and user have an increasing need to be mobile. To date, these two requirements have been met by two very different solutions, namely the Internet and mobile networks, respectively. The convergence of both solutions is the ultimate goal of third generation systems.

Most future third generation services will be implemented over packet-based networks [UMTSFORUM]. Internet continues to grow rapidly and networks based on the Internet Protocol will become predominant in the future. The Internet is clearly positioning itself as the single most important channel for delivery of information to the mass-market user.

Figure 3.1 on the following page gives an overview of an UMTS architecture.

UMTS are developed both for packet-switched (PS) and circuit-switched

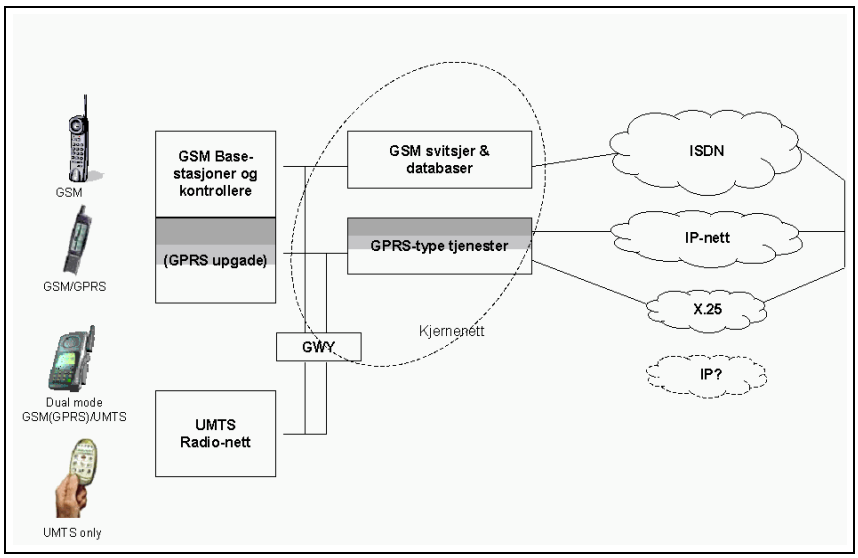


Figure 3.1: Overview of an OSA UMTS architectural issues [DIP]

(CS) networks. UMTS CS will be deployed soon, and will be combined with GPRS. Multimedia Telephony will then run over UMTS CS. UMTS CS is an interesting topic, but not for call related services due to the use of the H.324M standard for Multimedia Telephony. H.324M is a 3GPP protocol standard for 3G mobile multimedia services, as conversational and streaming video, over circuit switched systems. UMTS PS is a further development of the GPRS technology, and multimedia Telephony will then use SIP as a communication protocol [DIP]. Figure 3.2 on the next page and 3.3 on the facing page illustrates the 3GPP view.

3.2 Wireless Internet Network Communication Architecture

A new communications architecture that will enable true peer-to-peer wireless communication and client/server operation is being developed jointly by leaders in the mobile communications and Internet networking industries. The architecture described here is a hypothetical one, and bears resemblance to the 3GPP architecture for UMTS PS, IP based multimedia systems. Open Service Architecture (OSA) is the 3G service network, and handles services from 2G, PSTN, UMTS and IP, etc. I will describe OSA more closely in later chapters.

The resulting new architecture will deliver a world-class, IP-based networking solution to wireless network operators worldwide. The architecture supports current wireless standards and assists operators in their migration efforts to the emerging 3G wireless systems.

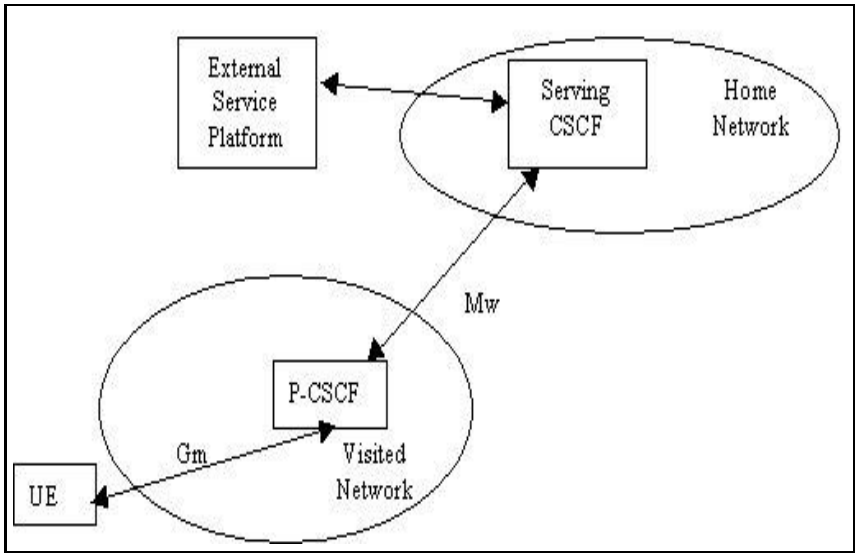


Figure 3.2: UMTS PS and IM subsystem from [3GPP]

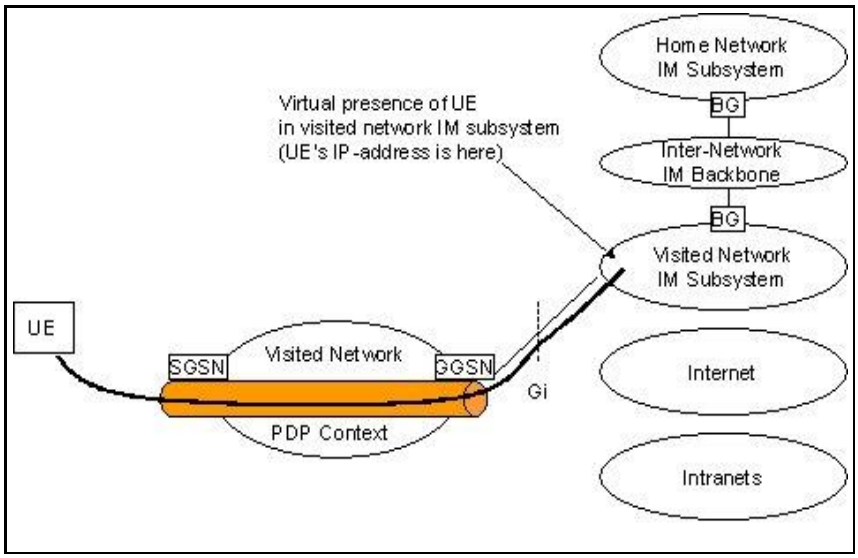


Figure 3.3: IM subsystem from [3GPP]

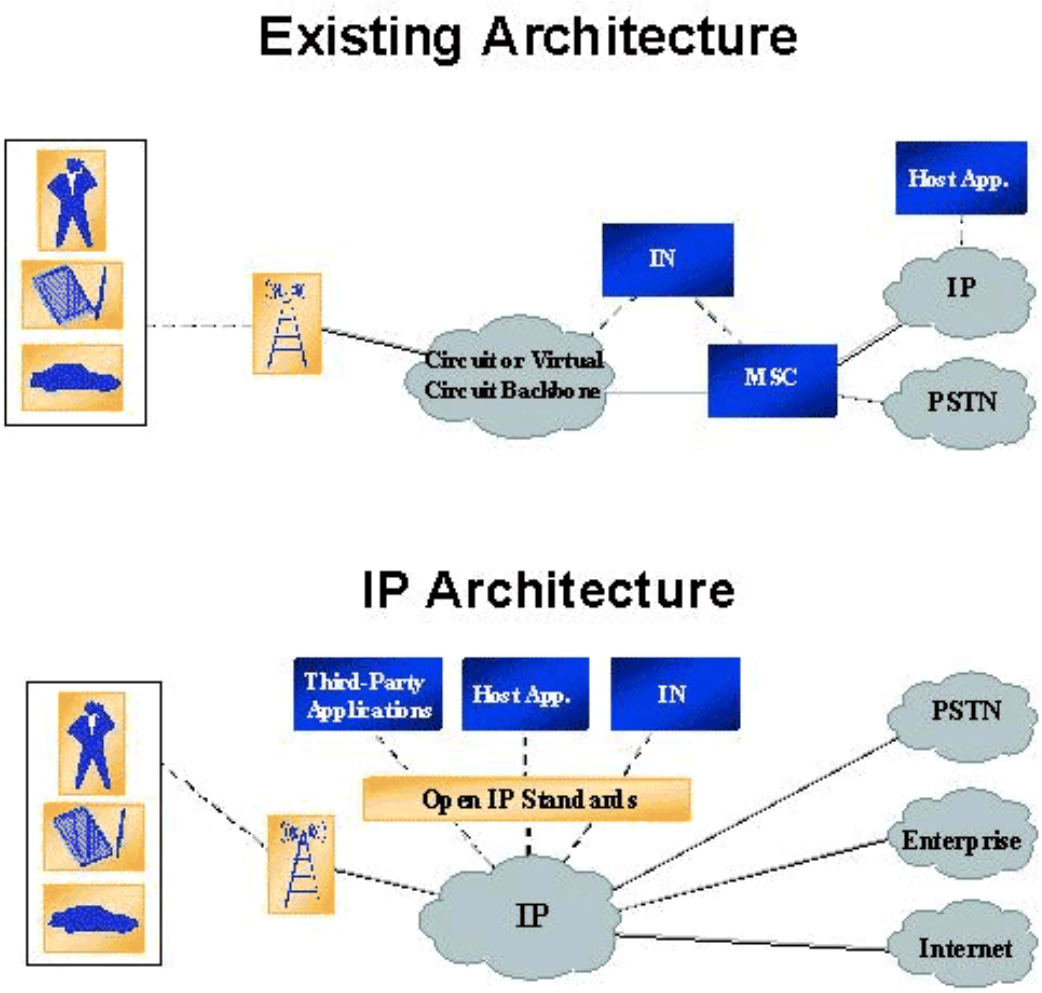


Figure 3.4: Overview of existing and new IP architecture [IECW]

[IECW] has identified several key deliverables that the wireless IP network of the future must provide. One of these are superior end-user experience.The network must be customizable, easy to use, and allow fast access to all services from a single device.

The fundamental premise driving the new communications architecture is the convergence of the voice and data realms. However, there are basic differences in the traditional regulation and provisioning of these services. In this model, the core network plays a central role in connecting a myriad of host services to subscriber access networks.

The architecture permits services to drive the functionality and added value of the wireless network. A key set of common functions and basic services must be supported to have a viable service offering. This key set of functions and services includes the ability for subscriber terminals to register, originate, receive, maintain (i.e., hand off or hand over), and clear voice and data sessions via one or more different media types. It also includes an initial set of supplementary services that add features to these basic call capabilities. The set of services offered will be expanded as value-added services, developed by third parties, and added to the network through open interfaces.

The common set of services enabled by the architecture is as follows:

- packet services-This is the data transport, providing carriage for services such as voice and data, that facilitates the delivery of wireless services. IP services include transparent support of existing IP applications and new value-added IP services that specifically leverage the advantages of a mobile environment.
- common functions-This is the base layer of functions, such as call registration and origination, that supports all subsequent layers of services and upon which all other layers of functions are based.
- teleservices-These are the basic services delivered to the end user, such as voice calls and data sessions.
- supplementary services- These are the enhanced teleservices, such as caller ID and teleconferencing, plus new supplementary services enabled by an IP-based services model.
- operations, administration, maintenance, and provisioning (OAM&P)-These are the supervisory functions, such as network monitoring and billing tracking, that keep the network operating efficiently and effectively.
- value-added services - These are the services residing outside the alliance architectures core network offering that are supplied by the network operator and content provider. See Figure **??** on page ??.

Chapter 4

IP Telephony

IP Telephony is the technology that allows transforming any IP network into a media for real-time transfer of long-distance and international voice calls and faxes.

The emergence of voice-over-IP (VoIP) applications and IP telephony has paved the way for a revision of the H.323 specification. The absence of a standard for voice over IP resulted in products that were incompatible. With the development of VoIP, new requirements emerged, such as providing communication between a PC-based phone and a phone on a traditional switched circuit network (SCN). Such requirements forced the need for a standard for IP telephony.

H.323 is an ITU-T standard with offspring in the traditional telecom world, and is based on ordinary signaling protocols like q.931 and ASN. SIP on the other hand has its offspring in the data world, and is therefore the IETF standard. SIP is a text-based protocol with a generic INVITE message. With newer versions of H.323 Q.931 messages are almost as fast in setting up a call as SIP. Although the standards is built on quite different foundations, SIP is simple and fast, and H.323 is complex and slower, the two protocol is now closing in on each other adopting the necessary functionality from each other. That is; H.323 is getting faster, and SIP is getting more functionality.

Since the work done in this paper is based on a H.323 system, SIP will not be further described.

4.1 The H.323 standard

The International Telecommunication Union - Telecommunication Sector (ITU-T) H.323 series of recommendations describes terminals, equipment, and services for multimedia communication over packet-based networks. [KOR]. The H.323 specification was approved in 1996 by the

ITU-T Study Group 16. Version 2 was approved in January 1998. The standard is broad in scope and includes both stand-alone devices and embedded personal computer technology as well as point-to-point and multipoint conferences. H.323 also addresses call control, multimedia management, and bandwidth management as well as interfaces between LANs and other networks [Dat98]. Many new enhancements have been introduced into the market-leading VoIP protocol H.323. Version 4 were approved November 17, 2000 and contains enhancements in a number of important areas, including reliability, mobility, and flexibility.

4.1.1 Protocols

The series of recommendation covers the protocols necessary for operation and for interconnection with circuit-switched networks. Call signaling is the messages and procedures used to establish a call, request changes in bandwidth of the call, get status of the endpoints in the call, and disconnect the call.

I will briefly describe some of the protocols. The description is based on [KOR] and [TOG].

- H.225.0 defines signaling between two endpoints, for example client and client, client and conference server, client and gateway, client and feature server, for basic call setup and release. This signaling is usually routed through a gatekeeper and is derived from Q.931 ISDN by using a subset of the Q.931 message and adding functionality in the user fields of the H.225.0 message. By using Q.931 as its basis for establishing a connection, H.323 allows for relatively easy bridging to the public switched telephone network (PSTN) and circuit-based phones. [TOG]
- H.225.0 registration, admission and status (RAS) defines signaling between the client and the gatekeeper for discovery of and registration to a gatekeeper by a client. H.225.0 RAS is also used by an endpoint to receive admission from a gatekeeper to place or receive a call.
- H.245 defines signaling between endpoints to exchange capabilities and carry end-to-end control messages governing the operation of the H.323 entity. The endpoints have a variety of capabilities, but the call established uses only the capabilities that are common to both endpoints. This signaling is routed through the gatekeeper.
- H.235 defines signaling for authentication of users and entities and encryption of data.
- H.450 series defines signaling between endpoints for supplementary services. This signaling is routed through the gatekeeper.
- RTP/RTCP for sequencing audio and video packets

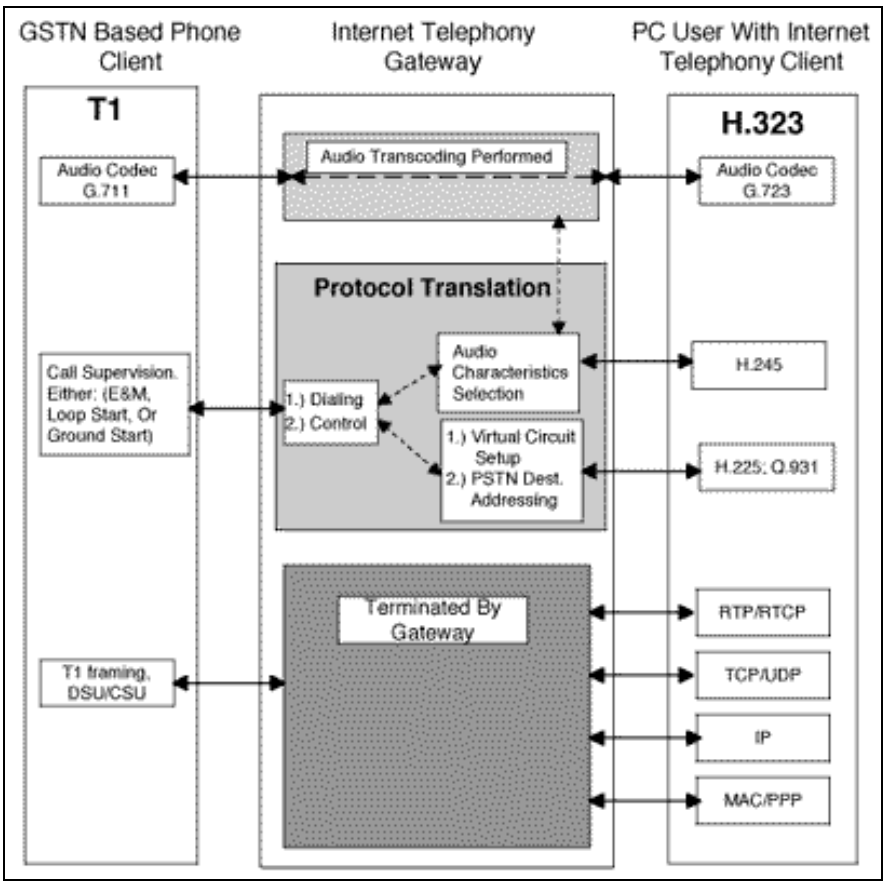


Figure 4.1: Sequence of VoIP connection [IECIP]

H.323 terminals must also support the G.711 audio CODEC. Optional components in an H.323 terminal are video CODECs, T.120 data-conferencing protocols, and conference bridge capabilities.

4.1.2 Architectural issues

A call goes over the local PSTN network to the nearest gateway server, which digitizes the analog voice signal, compresses it into IP packets, and moves it onto the Internet for transport to a gateway at the receiving end 4.1. With its support for computer-to-telephone calls, telephone-to-computer calls and telephone-to-telephone calls, Internet telephony represents a significant step toward the integration of voice and data networks.

The scope of H.323 does not include the LAN itself or the transport layer that may be used to connect various LANs. Only elements needed for interaction with the Switched Circuit Network (SCN) are within the scope

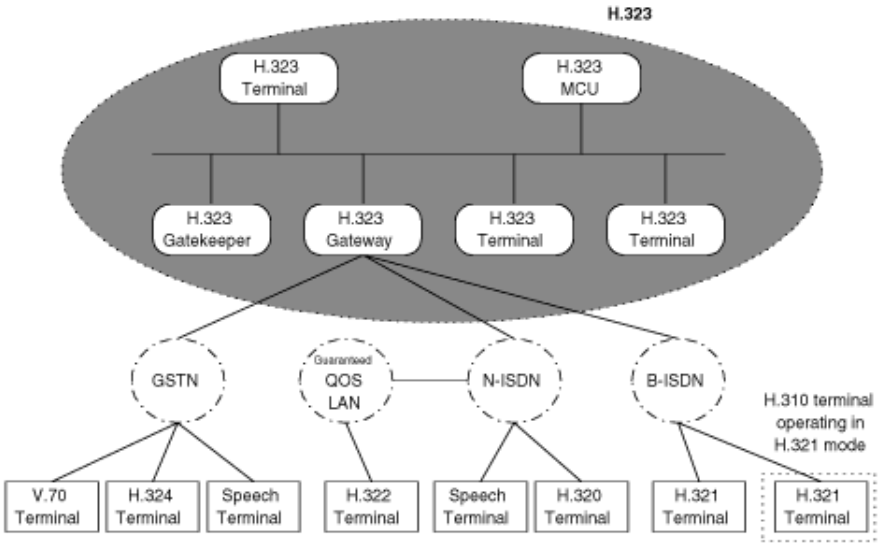


Figure 4.2: Overview of architectural issues for H.323 standard from [ITU2000]

of H.323. .

H.323 defines four major components for a network-based communications system: Terminals, Gateways, Gatekeepers, and Multipoint Control Units. Figure 4.2 outlines an H.323 system and its components [**?**].

Terminals are the client endpoints on the LAN that provide real-time, two-way communications. Figure 4.3 on the facing page describes the terminal components. All terminals must support voice communications; video and data are optional. H.323 specifies the modes of operation required for different audio, video, and/or data terminals to work together. It is, together with SIP, the dominant standard of the next generation of Internet phones, audio conferencing terminals, and video conferencing technologies.

All H.323 terminals must also support H.245, which is used to negotiate channel usage and capabilities. Three other components are required: Q.931 for call signaling and call setup, a component called Registration/Admission/Status (RAS), which is a protocol used to communicate with a Gatekeeper; and support for RTP/RTCP for sequencing audio and video packets.Figure 4.3 on the next page shows the architecture of a H.323 terminal:

Optional components in an H.323 terminal are video codecs, T.120 data conferencing protocols, and MCU capabilities

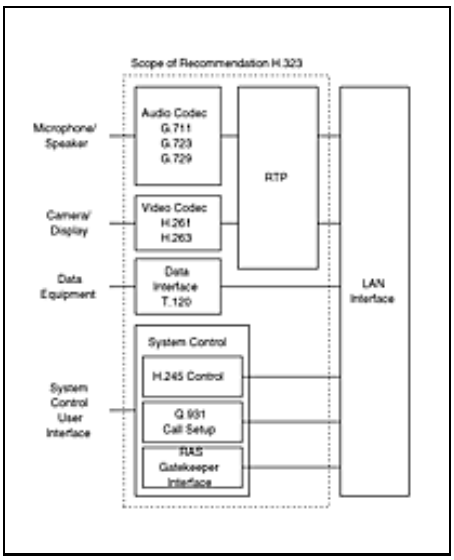


Figure 4.3: Overview of architectural terminal issues for H.323 standard from [Dat98]

4.1.3 Basic call completion in H.323

The figure 4.4 on the following page shows a basic call between a H.323 endpoint and a GSM, ISDN or PSTN terminal. When client A sends a Q.931 setup message to its gatekeeper X with the alias address of client B, gatekeeper X uses its address resolution to find the IP address of gatekeeper Y. Gatekeeper X then forwards the message to gatekeeper Y, which then sends the message to client B. If the user at client b answers the call, client B returns a Q.931 connect message via the two Gatekeepers to start the accounting. The connect message triggers the gatekeeper to the accounting for the call.

The figure shows how a basic call is made from client A to client B. It is assumed that the client A and the gateway representing client B have discovered and registered with their respective Gatekeepers, X and Y by using the H.225 RAS messages. It is also assumed that through inter-domain signaling, gatekeeper X knows that gatekeeper will complete calls for client B. Figure 4.5 on page 33 gives an overview of the H.323 messages including the H.245 parts..

4.1.4 Call Sequence in H.323

In H.323 there are a number of basic supplementary services. Since there have been a history of experience gained by the telephony industry in identifying valuable services, a large subset of these services have been

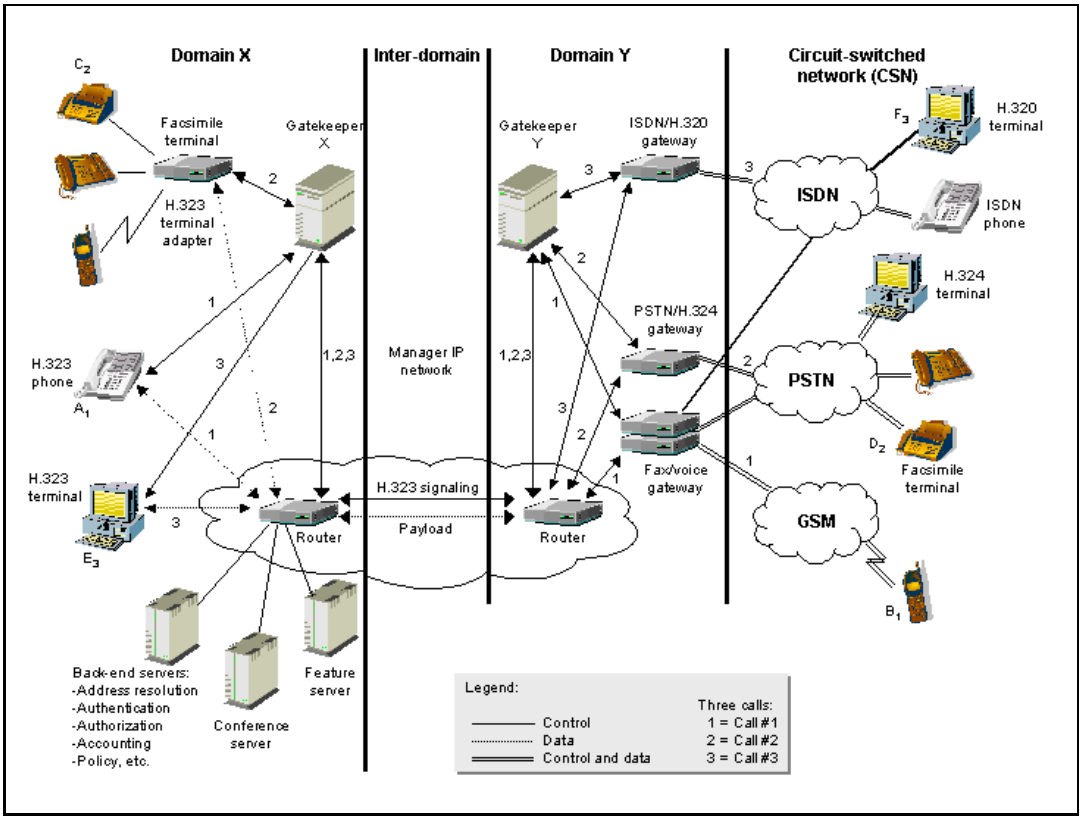


Figure 4.4: Overview of a basic call completion in an hypothetical H.323 network [KOR]

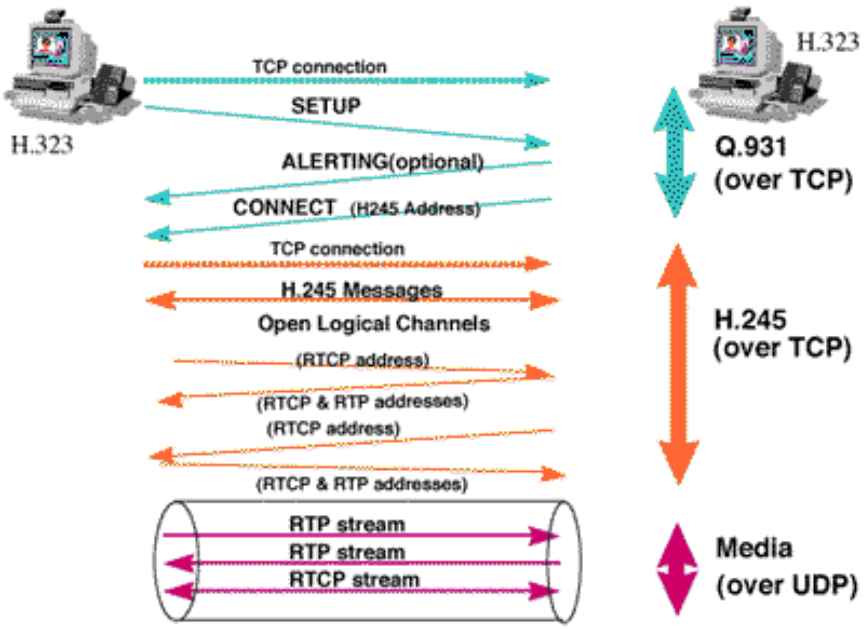


Figure 4.5: H.323 call sequence [IECIP]

specified as basic services in the H.323 and more precisely in the H.450.3 series of recommendations. One of these are the call forwarding service, also known as call diversion. It comprises the services call forwarding unconditional, call forwarding busy, call forwarding no reply and call deflection. It applies during call establishment by providing a diversion of an incoming call to another destination alias address. Any of the above variants of call forwarding may operate on all calls, or selectively on calls fulfilling special conditions programmed or manually selected by the user.

In traditional telephony, a user is provided with only a few choices by which a call can be forwarded. But in H.323, forwarding can be made dependent on a variety of conditions, such as the state of the called party, busy, no reply, absent, the caller identification; the time of day or day of week [KOR].

4.1.5 H.323 future features

Some new features and correction has been made in the latter version of the H.323 standard. The correction to FastStart and FastConnect handling makes improvements to the slow opening of media.

The added models for supplementary services introduce a set off brand new features. Since this is directly concern with the annex K I will describe this is more detail.

The ability to support a large variety of supplementary features and services is a requirement for many telephone solutions, regardless of the underlying technology. For many such services, an associated requirement is that a high level of interoperability exists between equipment provided by different vendors. The goal, therefore, is to define a standard that is sufficiently flexible that it can support all, or most of, the service that a vendor may wish to supply.

Within an H.323 environment, there are several different methods by which services can be provided; the H.450 series of recommendations, H.248 in association with its packages, Annex L/H.323, and Annex K/H.323. Although there is commonality of certain design goals for each of these solutions, the emphasis varies and each is more appropriate for certain circumstances. These solutions represent a spectrum of options for system and feature implementation. From purely peer-peer functionality control to purely client-server stimulus control, using either first or third party control. Rather then competing, they are complementary, allowing freedom of choice to the system developer.

The H.450 series of recommendation is designed for interoperability of services at a functional level. Its derivation from QSIG ensures interworking with many private networking systems. Services are defined for peer-peer relationships, with feature intelligence typically resident in the endpoint. An H.450 based service must normally be explicitly supported by each affected endpoint in the system. This distribution of service control allows endpoints to be more self-supporting and self-contained and is ideally supported by higher-end points. (Terminal centric, demand updates to the endpoint, if not backwards compatibility in standard, or loss of service subscription)

The other protocols provide for stimulus level control, where a full understanding of a service is normally only required by a single entity, typically in a master-slave relationship. Such stimulus-based methods use a set of well-defined atomic functions, which in various combinations provide any number of services. Stimulus protocols simplify the introduction of new services. However different implementations of the same service may differ sufficiently to complicate interoperability, even within the same network type.

Annex K/H.323 allows third party control of an H.323 call based on a separate control channel, using HTTP for user interaction. This party may be a service node within the operator domain or an external third party, representing a third party service provider. There is no fixed set of capabilities for the user interface, as various types of text formats, images and sounds may be utilized dynamically as registered MIME types. The service provider, the HTTP server, is responsible for the mapping between HTTP events and call control action, H.450 or other messages, for supplementary services, so the H.323 endpoint is unaware of the H.323 application. The service provider may be associated with the local Gatekeeper, the remote endpoint, or the remote gatekeeper within a call.

The URL scheme is introduced in Version 4 of the protocol. The H.323 URL will allow entities to access users and services in a consistent manner, much like other defined URLs allow for other IP-based services. The form of the H.323 URL is "h323:user@host", where "user" is a user or service and "host" might be the Gatekeeper that can translate the URL into a call signaling address [Dat98]. One of the alias types defined by H.323 is the url-ID, which is intended to contain standard URL schemes that may be used to reach resources. In addition, URLs are within Annex K/H.323 and may be used in other services added to H.323 in the future. An H.323 entity may accept any valid URL that it understands, but should support the H.323 URL as defined in this section. The following is based on [ITU2000]

The H.323 URL is intended to help an entity resolve the address of another H.323 entity. It is composed of two parts: the user and the hostport. The user specifies an alias for the entity, such as a user endpoint or a service, without carrying any information about the location of the entity. The hostport, on the other hand, may specify the transport address of a Gatekeeper, a Gatekeeper Identifier, or the fully qualified domain name of the Gatekeeper, along with an optional port number for RAS communication.

The H.323 URL is defined in ABNF as shown below.

```
H323-URL    =       "H323:" [ user ] [ "@" hostport ]
user        =       unreserved / escaped
hostport    =       host [ ":" port]
host        =       hostname / IPv4address / IPv6reference
hostname    =       *( domainlabel "." ) toplabel [ "." ]
domainlabel =       alphanum / alphanum *( alphanum / "-" ) alphanum
toplabel    =       ALPHA / ALPHA *( alphanum / "-" ) alphanum
IPv4address =       1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT
IPv6reference =     "[" IPv6address "]"
IPV6address =       hexpart [ ":" IPv4address ]
hexpart     =       hexseq / hexseq "::" [ hexseq ] / "::" [ hexseq ]
hexseq      =       hex4 *( ":" hex4)
hex4        =       1*4HEXDIG
port        =       1*DIGIT
unreserved  =       alphanum / mark
alphanum    =       ALPHA / DIGIT
mark        =       "-" / "_" / "." / "!" / "~" / "*" / "'"
escaped     =       "%" HEXDIG HEXDIG
```

As indicated in the syntax, the user string and the hostport string are both optional, though at least one shall be present.

The H.323 URL is not case sensitive. If an endpoint registers with a Gatekeeper and does not provide a hostport string, the Gatekeeper may append a hostport string to the URL when it returns the aliases in an RCF message. The endpoint shall accept the modified alias and use it when

sending subsequent requests to the Gatekeeper.

One of the primary goals in the development of the H.323 standard was interoperability with other multimedia-services networks [**?**]. This interoperability is achieved through the use of a gateway. A gateway performs any network or signaling translation required for interoperability.

4.2 IP Telephony and Mobility

As IP Telephony has seen a tremendous surge in interest in the last couple of years, mobility and wireless issues have not been considered in detail within the scope of IP Telephony until recent. The focus has mainly been on fixed IP telephony systems. I will point to some issues that need to be considered when mobility and wireless transport is introduced within IP Telephony [ITUMO].

Today, there exist several protocols that aid in delivering the promise of IP Telephony - reduced cost, grater flexibility, better manage-ability and enchanted service. While mobility has been considered to some extent within SIP, it has not yet been addressed comprehensively within H.323 and MGCP. It is imperative that the mobility issues surrounding IP Telephony be understood and suitable solutions in order to access those issues.

According to [ITUMO] Mobility can be broadly of three types - terminal mobility, user mobility and service mobility - and provision needs to be made for each of these mobility types within an IP Telephony system. In addition, the circumstances under which one or more of these mobility types plays a role needs to be considered. For instance, the case of a user moving within his H.323 zone is very different from that in which the user moves from one zone to another. In addition, in many cases, the IP Telephony terminal may have a wireless access to the network. The design of an IP Telephony system needs to consider issues such as the bandwidth constraints of wireless access.

4.2.1 Terminal mobility

Still following [ITUMO] terminal mobility is broadly of two kinds: Discrete terminal mobility is the mobility of a terminal when there is no call in progress. A user may or may not be registered at the terminal. Continuous terminal mobility is the mobility of a terminal while the user is registered at a terminal and a call is in progress. Continuous terminal mobility is concerned with maintaining session connectivity under such a scenario.

Discrete as well as continuous mobility may occur either at the link layer only or at the network layer as well. Link layer mobility techniques are responsible for tracking the movement of the terminal from one point of attachment to another, thereby facilitating seamless network access.

When the terminal movement is restricted to the link layer, then link layer mobility techniques suffice. On the other hand, terminal mobility could result in a change in the sub-net that the terminal is attached to, thereby resulting in mobility at the network layer as well. When mobility occurs at the IP layer, that is the network layer in question, Mobile IP along with link layer mobility techniques suffice. In a cellular environment, hand-off usually involves link layer mobility while roaming involves network layer mobility.

The existing link level technologies, cellular, bluetooth, infrared Data Association and Wireless LAN makes it possible for a terminal to move from the coverage of one link-level type to another. A roaming terminal needs to understand the link-level technology before it can provide useful service to a subscriber. Multi-band terminals that are capable of understanding multiple link-level technologies are needed for this purpose. Moreover, each of the link layer technologies has its own mechanism for QoS, security etc. Mechanisms for QoS and security also exists at the IP layer and may even be incorporated at the application layer. In this regard, some of the issues that need to be considered in an IP Telephony system are mapping of IP QoS to radio QoS, and consideration of which security mechanisms that are available in the different layers - link, IP or application - that needs to be turned on. Activating IP layer security mechanisms such as IPSec may obviate the need to activate any available link layer security mechanisms.

4.2.2 User mobility

User mobility deals with the mobility of a user prior to making a call; thereby making the user capable of using different equipment possibly connected to different networks. The user may also use more than one equipment simultaneously. Some issues - user authentication, user identification, user identifier portability; user registration, need to be considered when dealing with user mobility in an IP Telephony system. Closely related to user mobility is the concept of user identifier portability. This involves the user retaining his address identifier when the user changes geographical location, service subscription, including change of network type, for example from X.25 to IP, and other service providers. The preferable method for achieving such portability is to define two identifiers:

User identifier is the identifier that uniquely identifies a user. The identifier conveys no information regarding the users geographical location, service subscription or service provider. The home identifier, which may be dynamically assigned to a use, usually provides means to reach the user's home network. The home identifier may or may not be hidden from the end-user. Readability of the user identifier is critical to make it easily transportable via non-electric means. Additionally, directories are needed to allow users to publish their identifying information, that is name and address, and the user identifier, if they so desire. The authenticity of the published identifying information must be assured, so that the

user identity can not be hijacked. The user should control the use of the identifying information, so that this could not be listed in directories he does not want to be published in.

The mapping between user identifier and home identifier is stores in a suitable database or directory. The form of the user identifier should be hierarchical to facilitate ease of directory lookup. With user identifier portability, when a user changes providers or service subscription, he retains his user identifier and the home identifier points to his new home network. Similar, when a user changes his geographical location on a "permanent" basis, he could again retain his user identifier with the home identifier being made to point to his new home network. In addition to the users home identifier pointing to the users home network, a mechanism is needed to handle the case where the user is currently visiting another network on a "temporary" basis, and needs to be reached. This is achieved when the user registers with the visiting network, and the users current location is conveyed to his home network.

The identifier that is used when the user registers with a visited network is referred to as the customer identifier, and it contains two parts: a global unique home network identifier and a user identifier within the home network that uniquely identifies the user within his home network.

When the user registers with a visited network, the customer identifier is usually sent in the clear since the user do not share a secret with the visited network. Using well-known identifiers for the customers' identifier could result in the user's current location being known to an eavesdropper. This must be looked into in an IP Telephony system, as well.

After the user has successfully registered with the visiting network, he is assigned a care-of-identifier by the visited network. This care-of-identifier is used by the home network to suitably route packets destined to the mobile user. For instance, in Mobile IP, the care-of-identifier is the care-of-address assigned to the mobile node.

When the user registers with a visited network, the user needs to be authenticated in a consistent fashion. Several authentication mechanisms exists, all of which can be divided into three categories: Something the user knows - password based schemes exists, all of which can be divided into three categories:

- Something the user knows - password based schemes
- Something the user has - a magnetic card or smart card
- Something the user is - biometrics (retina scan or thumb scan)

The most promising mechanism today seems to be a smart card based approach that also uses a password or pin. This covers the first two categories described. Using the password, a challenge-response mechanism could be used for authentication. Such authentication mechanisms need to

be tied into the signaling mechanism at application layer.

To better understand what this means for an IP Telephony system, I will briefly consider the instantiation of the identifiers mention above in a few existing systems:

In the Public Switched Telephone Network (PSTN) the user identifier is the name of the individual or company along with the city and country that the individual user or company is resident in. Using a directory service, this user identifier can be mapped to a home identifier that is the E.164 number. Mobility in PSTN usually occurs in the form of calls to the particular telephone number being forwarded either temporarily or permanently to a different telephone number. In order to forward calls, the user usually identifies himself to the service provider by dialing his service provider and, say, his telephone number and billing address. Thus, the user's telephone number is also his customer identifier while the billing address, or some password, is used for authentication. The care-of-address is the new E.164 number that the call is forwarded to. In the Global System for Mobile Communication (GSM) the user identifier is, as in the case of PSTN, the name of the individual user or company along with the city and country that the individual user or company is resident in. Using a directory service, this is mapped to a home identifier, which is the MS-ISDN. The customer identifier is the IMSI, and a challenge-response mechanism based on a shared secret stored in the user's SIM-card and Authentication Center (AuC) is used to authenticate the user.

In IP access, the home identifier is usually the user's email address. There is no clear user identifier in this case - if one existed, then a directory service would provide the mapping from the global unique user identifier to the home identifier, email address. During IP roaming, the user identifies himself to the home network, using Network Access Identifier (NAI), which is the customer identifier. Usually, a challenge-response mechanism using a shared password is used to authenticate the roaming user. The home identifier is the IP address that is dynamically assigned to the user's terminal.

4.2.3 Service Mobility

Service mobility deals with the ability of a user to access a service that he ahas subscribed to with or without terminal and also user mobility. A Virtual Home Environment (VHE) is created when the user's entire service set is transparently available to the user irrespective of his current location.

Three possible mechanisms exists to achieve service mobility:

- Upon registration in a visited network, the users entire service set is down loaded to the visited network.
- The user's service set is not down loaded within a visited network.

Instead, each time the user wishes to access a certain service, the home network is contacted.

- A hybrid case, in which a portion of the user's service set is down loaded in a visited network, and to access any of the remaining services, the home network needs to be contacted.

There is trade-off between downloading and not downloading the user's service profile. The former is preferred when the user is relatively static, when the home gatekeeper trusts the visited gatekeeper sufficiently to transfer the user's set of service to it, and when optimized routing for call signaling is desired. The latter is preferred when the home gatekeeper wishes to keep a tight leash on the calls made by its subscribers and when the home gatekeeper does not whish to down load a subscribers service set to a visited gatekeeper. There are two possible mechanisms by whish the user's service set may be down loaded in a visited network:

The user carries the service profile with him, e.g. in a SIM card, signed by the home network provider. Then the mobility aspect vanishes. Or, contacting the home network gatekeeper to down load the users service profile.

Services may be broadly classified into two: transport services and application services. Transport services refers to the services that the network offers to the user - for intense network QoS and security as mentioned. Application services refer to services that the user is offered at the application level. This includes features such as call waiting, call forwarding, conference call etc.

With IP Telephony, service mobility involves transport service transparency as well as application-level service transparency. In this model, a user's profile is divided into a transport service profile and an application level profile. Clearly, both service profiles should be supported in a flexible fashion for a roaming user.

For user mobility as well as service mobility, the user needs to register with an entity, say a gatekeeper, in the visited network. Then the user needs to be aware that the user is in another network than his home network, and the user needs to discover a suitable gatekeeper in the visited network to register with. Several mechanisms exist to achieve this. For instance, the terminal may realize that it is a new visit network when it stops receiving advertisements from the gatekeeper that it is registered with. This may be the case when the gatekeeper multicasts advertisements with a limited scope. Similar, mechanisms used to discover a gatekeeper include the following procedures: Active probing by a terminal using suitable multicasts mechanisms Passive listening of the terminal to gatekeeper advertisement.

Chapter 5

H.323 Annex K

The new Annex K describes a means of providing HTTP-based control for H.323 devices. With this annex, service providers have the ability to display web pages to the user with meaningful content that ties into the H.323 systems. In essence, it is a third party call control mechanism that utilizes a separate HTTP connection for control. This should not be confused with the simple ability to redirect a user to a web page– something that H.323 has been able to do since Version 2. Rather, this is a new "service creation" environment, which is unlike anything described to date. Basic mechanisms is standardized in H.323 version 4, but there is not a need for standardizing new services. New features may be introduced without the delay introduced by any formal standardization process [Dat98].

The core of the Annex K service is to provide an optional way for controlling user directed supplementary services in an H.323 environment, for example the IP telephony system. By opening a seperat connection conveying a service independent control protocol, new services may be developed and deployed without updates to the H.323 endpoints, that is the terminals holding the H.323 endpoints [ITUTD-13].

The data exchanged on this channel is meant to be informative and should be followed by appropriate actions, for example H.450 invocations or invocation of other call related supplementary services. The service control channel may be utilized for both call related and non-call related services. It may be opened between the terminal and the network, or between two endpoints. Annex K supports peer-to-peer call independent signaling, and also the use of the Hypertext Transfer Protocol (HTTP). Figure 5.1 on the following page gives an overview for the HTTP based service control. In addition to call-related service control this annex also gives opportunity for non-call related service control. I will not describe this further in this paper.

It is generally up to the provider of the URL to define and implement the control functions and services that are being presented by the given URL. Both standardized and non-standardized services may be supported.

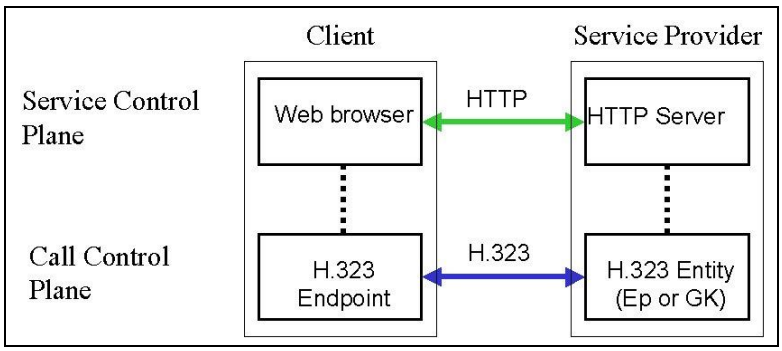


Figure 5.1: Annex K system overview for HTTP based service control from [ITUTD-13]

If the service interact with H.323 call processing, that is the cannel is being used in a call between two endpoints, the provider of this URL should make the binding between the HTTP service and the H.323/H.450 services that are being supported by the gatekeeper or endpoint. Any application that utilizes this service control channel should consider any possible service interaction into consideration.

5.1 Use of H.323 in Annex K

H.323 messages are used for maintaining the service control sessions. A service control session is a one way relationship between the client entity and the service provider [ITUTD-13]. It is initiated from the client after the receipt of a *ServiceControlAdress* URL in H.225 messages. The URL may be received through two different H.323 signaling channels:

- A *ServiceControlSession* structure containing a URL is received over the RAS channel
- A *ServiceControlSession* structure containing a URL is received in a message on the H.225.0 call signaling message.

Any service provider wishing to initiate a new service control session does so by sending a *ServiceControlSession* structure to the client. It contains a sessionId, to identify this particular session, since several sessions could be sent to the client at the same time by different service providers not aware of each other, and the URL for the service. The client may open a connection to this address and request the resources from the URL. The client, or more, the user, can end the session at any time, without any notification to the service provider.

For opening a service control session related to a specific call, a service control session are opened between an endpoint and a gatekeeper or between two endpoints with a *ServiceControlSession* field containing a URL in the call signaling message.

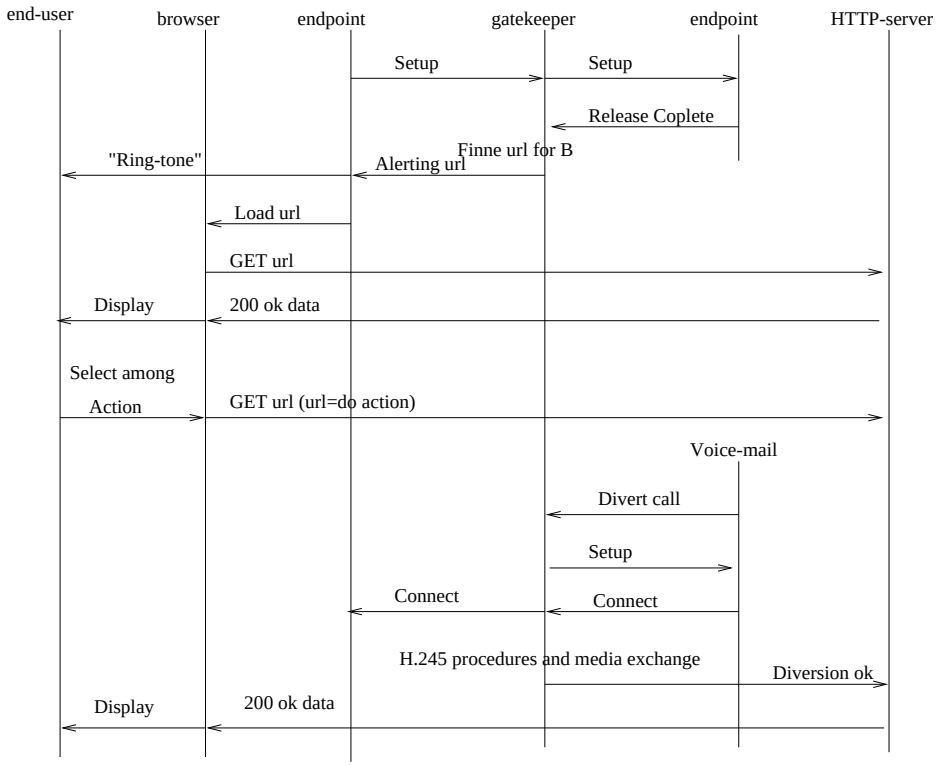


Figure 5.2: Call related service control, gatekeeper routed call based on [ITUTD-13]

5.2 Use of HTTP in Annex K

In order to support call-related service control a URL is conveyed in different H.225.0 messages, for example alerting and release complete. An endpoint when it receives such an URL should request a standard HTTP user agent, that is normally a web-browser, to open and render that URL. The HTTP user-agent should render the given URL and support style-sheets, scripts, links and images according to that defined in [IETF97]. Actions defined and executed by the contents of this URL could be executed locally, with for example mail-to links as in e-mail options, or remote on any linked HTTP server, that is being implemented or related to an endpoint or a gatekeeper. Alas, both the endpoint or the gatekeeper can be the service provider. In this paper I will focus on the gatekeeper as the service provider.

Figure 5.2 illustrates a variation of a “Call waiting service” with options for the calling party.

The gatekeeper detects that the calling party is busy. In the solution implemented in this paper the same approach is used, but the gatekeeper do not detect busy, instead the gatekeeper is being notified from the Service Agent that the called end-point do not want to answer the initiated call, and the gatekeeper will read this as busy. Upon this, the gatekeeper provides an URL to the calling party. The gatekeeper have received this URL from the Service Agent, that has identified this URl during the execution of the service logic. The URl reference a Web page containing a set of options for the further processing of the call.

The user hears the audio alert. This audio alert should be optional, and other means for altering should be possible to choose among. Further more the user should be able to set the configuration for how the presentation of the web-page should be handled. Several possible action may be desired; an automatic opening of an web-page, the user must active and open the web-page, or the user can end the session without even open the web-page. These are questions for user interaction and interactive web-design. It should be considered for a service implementing this logical channel for call control.

When the web-page is presented different option for how to further process the call is presented the user. The option could be for example divert to voice-mail, e-mail or operator. The user selects the voice-mail and this selection is signaled to the HTTP-server that informs the gatekeeper about this.

The gatekeeper effectuates the diversion and informs the HTTP-server about the successful diversion. The HTTP server then responds to the browser with a new web-page saying for example that the diversion was completed successfully and giving the user som new option.

Chapter 6

The Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) has been used in the World Wide Web global information initiative since 1990. The following description is based on specification worked out by the Internet Engineering Task Force (IETF) ended in September 1995. This specification reflects preferred usage of the protocol.

The Hypertext Transfer Protocol is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems [IETF97]. It is a generic, stateless, object-oriented protocol which can be used for many tasks, such as name servers and distributed object management systems, through extensions of its request methods (commands). A feature of the HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transfered.

6.1 Overall operation

The HTTP protocol is based on a request-response paradigm. A requesting program (termed a client) establishes a connection with a receiving program (termed a server) and sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content. The servers responds with a status line(including its protocol version and success or error code), followed by a MIME-like message containing server information, entity meta-information, and possibly body content. Any given program can be both client and server, the use of these terms refers only to the role being performed by the program during a particular connection, rather then the programs purpose in general.

On the Internet the communication generally takes place over a TCP/IP

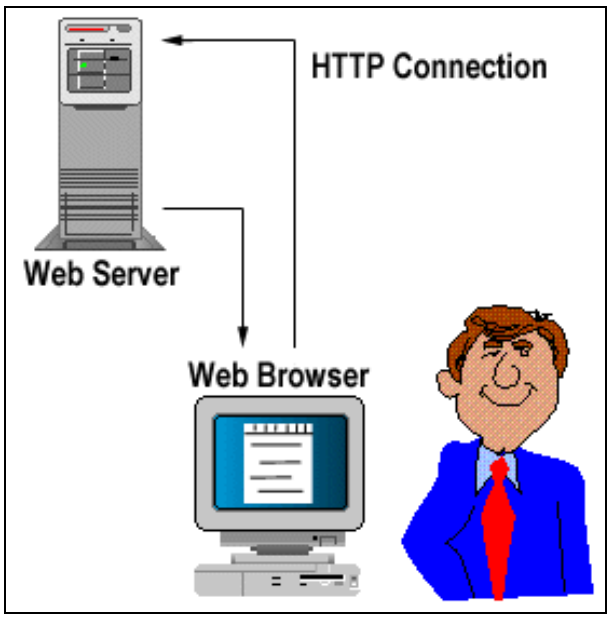


Figure 6.1: HTTP client/server overview from [BASHT]

connection. The HTTP protocol can be implemented on top on any other protocol on the Internet, or on any other networks [IETF97]. For most implementations the connection is established by the client prior to each request and closed by the server after sending response. However, this is not a feature of the protocol. Both clients and servers must be capable of handling cases where either party closes the connection prematurely, due to user actions, automated time-out, or program failure. In any case the closing of connection by either or both parties always terminates the current request, regardless of its status.

6.2 HTTP Header Fields

An HTTP transaction consists of a header followed optionally by an empty line and data. The header specifies action required of the server, the type of data being returned and a status code. The use of header fields sent in HTTP transactions gives the protocol flexibility. These header fields allow descriptive information to be sent in the transaction, enabling authentication, encryption, and user identification. The header is a block of data preceding the actual data, and is to as the meta-information, that is the data is about the information to come.

I will briefly describe some of the different header fields:

- The *Content-Type* header fields indicates the media type of the data sent to the recipient, or in the case of the HEAD method, the media type that would have been sent had the request been a GET. This field is used by browsers to know how to deal with the data. The client uses this information to determine how to handle a videofile or an inline graphic.
- The *Date* header represents the date and time at witch the message was originated.
- The *Expires* field gives the date after witch the information in the document ceases to be valid. Caching clients, including proxies, must not cache this copy of the resource beyond the date given, unless its status has been updated by a later check of the origin server.
- The *Form* header field, if given, should contain an Internet e-mail address fro the human user who controls the requesting user agent. This header field may be used for logging purposes and as means for identifying the source of invalid or unwanted requests. It should not be used as an insecure from of access protection. The interpretation of this field is that the request is been performed on behalf of the person given, who accepts responsibility for the method performed. In particular, robot agents should include this header so that the person responsible for running the robot can be contacted if problems occur on the receiving end.
- The *If-Modified-Since* header field is used with the GET method to make it conditional; if the requested resource has not been modified since the time specified in this field, a copy of the resource will not be returned from the server.
- The *Last-Modified* header field indicates the date and time which the sender believes the resources was last modified. This field is useful for clients that eliminate unnecessary transfer by using caching. The exact semantic of this field are defined in terms of how the user should interpret it; if the receiver has a copy of the resource which is older than the date given by the *Last-Modified* field, that copy should be considered stale.
- The *Location* response header field defines the exact location of the resource that was identified by the request. If the value is a full URL, the server returns a *redirect* to the client to retrieve the specified object directly. If instead the goal is to reference a file on the local server, only output a partial URL.
- The *Referer* request header field allows the client to specify, for the server's benefit, the address(URI) of the resource from which the request URI was obtained. This allows a server to generate lists of backlinks to resources for interest, logging and optimized caching. It also allows obsolete or mistyped links to be traced for maintenance.
- The *Server* response header field contains information about the software used by the origin server to handle the request. The field can

contain multiple product token and comments identifying the server and any significant subproducts. precisely

- the *user-agent* field contains information about the user agent originating the request. This is for statistical purpose, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations, such as inability to support HTML tables.

The following is an example of an header:

```
HTTP/1.1 200 Document follows
Date: Tue, 14 Apr 1997 09:25:19 PST
Server: JWS/1.1
 Last-modified: Mon, 17 Jun 1996 21:53:08 GMT
Content-type: text/html
Content-length: 4435

 <4435 bytes worth of data -- the document body>
```

6.3 Requests

The HTTP/1.0 allows an open-ended set of methods to be used to indicate the purpose of a request [IC99]. A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource requested, the identifier of the resource, and the protocol version in use. The request line begins with a method token, witch indicates the method to be performed on the resources identified [IETF97]. The methods available in HTTP\1.0 are:

- GET - means retrieving whatever information, in the form of an entity, that is identified by the request URI.
- HEAD - is identical to GET except that the server must not return any Entity-Body in the response.
- POST - is used to request that the destination server accepts the entity enclosed in the request for a new subordinate of the resource identified by the request URI.
- PUT - requests that the enclosed entity should be stored.
- DELETE - requests that the origin server delete the resource identified by the request URI.
- LINK - establishes one or more LINK relationship between the existing resource identified by the request URI and other resources.
- UNLINK - removes one or more LINK relationships from the existing resource identified by the request URI.

Since the implementation of the OnNoAnswer service uses the GET and POST methods I will describe these in more detail.

6.3.1 The GET method

The GET method is used to ask for a specific document [IC99]. More , the GET method means retrieving whatever information in the form of an entity is identified by the request URI. If this refers to a data-producing process, it is the product data witch shall be returned as the entity in the response and not the source text of the process. The semantic of the GET method changes to a conditional GET if the request message includes an *If-Modified-Since* header field. A conditional GET method requests that the identified resources shall be transfered only if it has been modified since the date given by the *If-Modified-Since* header. The conditional GET method is intended to reduce network usage by allowing cached entities to be refreshed without requiring multiple requests or transferring unnecessary data.

The algorithm for determining a conditional GET is:

- If the request would normally result in anything other than a “200 OK” status, or if the passed *If-Modified-Since* date is invalid, the response is exactly the same as for a normal GET.
- If the resources has been modified since the *If-Modified-Since* date, the response is exactly the same as for a normal GET.
- If the resource has not been modified since the *If-Modified-Since* date, the server shall return a “304 NOT MODIFIED” response.

The HEAD method, not used in the OnNoAnswer Service, is identical to GET except that the server must not return any Entity-Body in the response.

6.3.2 The POST method

The POST method is used to request that the destination server accepts the entity enclosed in the request as a new subordinate of the resources identified by the request URI in the request line. POST is designed to allow a uniform method to cover the following functions [IETF97]:

- Annotation of existing resources.
- Posting a message to a bulletin board, newsgroups, mailing-list, or similar group of articles.
- Providing a block of data, usually a form, to a datahandling process.
- Extending a database through an append operation.

The actual function performed by the POST method is determined by the server and is usually dependent on the request URI. The posted entity is considered to be subordinate to that URI in the same way that a file is subordinate to a directory containing it, a new article is subordinate to a newsgroup to which it is posted, or a record is subordinated to a database.

The client can suggest a URI for identifying new resources by including a URI header field in the request. However, the server should treat that URI as advisory and may store the entity under a different URI or without any URI.

The client may apply relationships between the new resource and other existing resources by including *Link* header field. The server may use the *Link* information to perform other operations as result of the new resource being added. For example, lists and indexes may be updated. However, no mandatory operations is imposed on the origin server. The origin server may also generate its own additional links to other resources.

A successful POST does not require that the entity be created as a resource on the origin server or made accessible for future references. That is, the action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either "200 OK" or "204 NO CONTENT" is the appropriate response status, depending on whether or not the response includes an entity that describes the result.

If a resource has been created on the origin server, the response should be "201 CREATED" and contain the allocated URI, all applicable *Link* header field, and an entity, preferably of type-content: "text/html", which describes the status of the request and refers to new resource.

A typical POST request might be as follows:

```
POST /servlet/MyServlet HTTP/1.1
User-Agent: Mozilla/4.0 (
  compatible;
  MSIE 4.01;
  Windows NT)
Host: www.magelang.com
Accept: image/gif, image/x-xbitmap,
image/jpeg, image/pjpeg, */
Content-type: application/x-www-form-urlencoded
Content-length: 39

 name=Scott&company=MageLang\verb,%,20Institute
```

Note the blank line–this signals the end of the POST request header and the beginning of the extended information.

6.4 CGI and servlets

I will in this chapter also introduce the concept of Servlets. I use Servlets in the prototype of the OnNoAnswer service.

Web separates the generation of content(which is the services) from the detailed protocol handling, using one of several possibilities; Common Gateway Interface (CGI), Java Servlets, or Server Side Java Script. [Ros99]

There are a number of mechanisms which provides “service” (that is dynamic content” in the WWW. These can be broken down into two types - a server side and a client side. Server side mechanisms allow the content of the web response to be generated by either a separate process (in the case of the CGI, or by the web server (in the case of server side Java Script, Java Servlets, or Active Server Pages (ASP)). In the case of client side programming, the web page returned by the server can contain programs which execute to either interact with the user, or provide dynamic content, for example Java and Java Scripts.

An HTTP servlet is a direct replacement for Common Gateway Interface (CGI) scripts. HTTP servlets are accessed by the user entering a URL in a browser or as the target of an HTML form action [SUN]. Servlets are not restricted to generating web pages; they can perform any other function, such as storing and fetching database information, or opening a socket to another machine.

A Servlet is a Java program that is run by a web server. Two of the main uses of servlets are to generate dynamic html documents and allow for client/server communication with Java applets. The web server maintains a Virtual Machine internally, or as a separate process that is communicates with to run the servlet. Servlets can be used to implement a 3-tier architecture where for example the client is on one host, the web server is on a second host and the database on a third host. This also allows for access to distributed database from servlets running in the web server.

Currently servlets are used to add functionality to HTTP web servers, but the technology could also be applied to other types of servers using other protocols, such as FTP for example. A Generic Servlet is intended to be the superclass for any servlet implementation, possible to subclass for adapting to protocol.

A Generic Servlet provides a method called service() that the designer of a servlet must override.

```
public class Example extends Generic Servlet {

  public void service(ServletRequest req, ServletResponse resp)
```

```
                    throws ServletException, IOException
{
  resp.setContentType(''text/plain'');
  ServletOutputStream out = resp.getOutputStream();
}
```

The service method has two parameters, a request and a response. The ServletRequest contains information about a request, and it is possible to do a getParameters() for retrieving information. The ServletResponse contains information about the response that the servlet is generating. For example is the getOutputStream ,method used to acquire the stream that the Servlet must write its response to. HTML might be written to the stream if the servlet is designed to produce output displayed in a browser. Other data might be written to the stream if the servlet is communication with a Java client.

Today the only subclass and therefore protocol supported is the HTTP Protocol, so in practice, the servlet implementation will be a subclass of the HTTP Servlet, which adds web server-specific functionality.

HTTP Servlet provides two methods that are commonly overwritten in the class, doGet() and doPost(). These methods are called when GETs or POSTs are sent to the servlet. These methods works like the GenericServlet service() method, but take a HttpServletRequest and a HttpServletResponse as parameters. The HttpServletRequest contains information about a request, for example methods for getting a header, retrieval of query string, URL, protocol, users and others, for all; the possibility to retrieve any HTTP information needed. The HttpServletResponse contains information about the response that the servlet is generating, for example a method for setting a header, error status and status information using provided constants.

The HttpServlet also offer methods for Delete, Options and Trace. I will not describe these methods further since they are not used in the implementation. As shown in the following diagram, the service() method interprets each HTTP method and determines if it is an HTTP GET, HTTP POST, HTTP HEAD, or other HTTP protocol method.

When making HTML Servlet choices, it is good design to use a HttpServlet only when there is need for the added functionality, or to use both GET and POST methods. Browsers use the GET method for any URLs directly typed in, so if this is the case the GET method should be overwritten.

For the OnNoAnswer service this is not the case, initially. The Servlet is being called in by the plugin implementing the logic of the server. Anyway, for testing and prototyping the service this was the means of doing it, so the GET method was implemented.

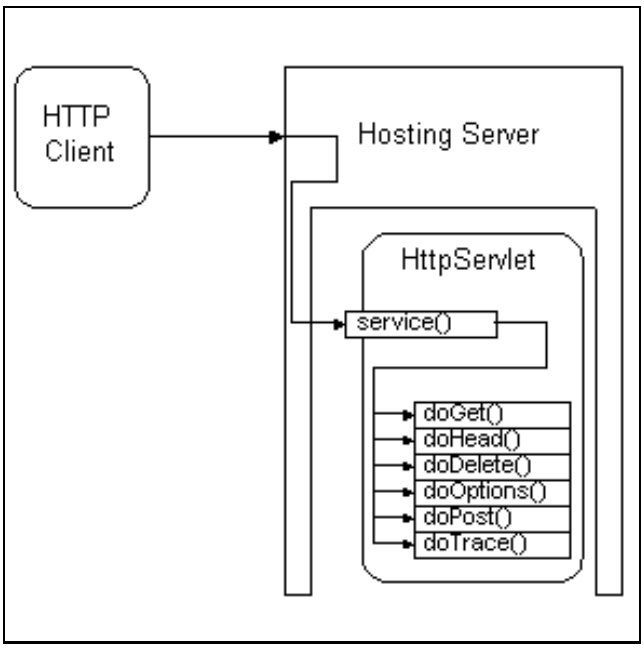


Figure 6.2: HttpServlet interaction between client and server from [IETF97]

```
public void doGet(HttpServletRequest req, HttpServletResponse resp)
                  throws ServletException, IOException
     {
       /* this could for further development be xml*/
       resp.setContentType(‘‘html/text’’);
       ServletOutputStream out = resp.getOutputSTream();
       .....

     }
```

Either GET or POST may be used from an HTML form. In general it is advised to use GET method when the servlet won't be making any changes to a resource such as a database. A POST method should be used in the opposite case, that is, when the servlet will be modifying a resource. It is also good practise to set the desired header fields before writing to the output stream.

```
public void doPost(HttpServletRequest req, HttpServletResponse resp)
                  throws ServletException, IOException
     {
       /* this could for further development be xml*/
       resp.setHeaderContentType(‘‘Content-Type, ‘‘html/text’’);
       ServletOutputStream out = resp.getOutputSTream();
```

```
    .....
}
```

6.5 The Hypertext Transfer Protocol - Next generation

The essential simplicity of HTTP has been a major factor in its rapid adaption, but this very simplicity has become its main drawback. The next generation Hyper Transfer Protocol, HTTP-NG, will be a replacement for HTTP/1.0 with much higher performance and adding some extra features needed for use in commercial applications. It is designed to make it easy to implement the basic functionality needed by all browsers, while making the addition of one powerful features such as security and authentication much simpler [IC99].

The current HTTP standard often causes performance problems on the server side, and on the network, since it sets up a new connection for every request. Simon Spero [IC99] has published a progress report on HTTP-NG.

> HTTP-NG..divides up the connection[between client and server] into lots of different channels...each object is returned over its own channel.

HTTP-NG allows many different requests to be sent over a single connection. These request are asynchronous, meaning that there is no need for the client to wait for a response before sending out a new request. The server can also respond to requests in any order it sees fit. It can even inter-weave the data from multiple objects, allowing several images to be transferred in parallel.

To make these streams easy to work with, HTTP-NG sends all its messages and data using a “session layer”. This divides the connection up into lots of different channels. HTTP-NG sends all control messages, like GET requests and meta-information, over a control channel. Each object is returned over its own channel. This also makes redirection much more powerful, for example; if the object is a video the server can return the meta-information over the same connection, together with a URL pointing to a dedicated video transfer protocol that will fetch the data for the relevant object. This becomes very important when working with multimedia aware networking technologies. The HTTP-NG protocol will permit complex data types such as video to redirect to the URL to a video transfer protocol, and only then will the data be fetched for the client. Currently the W3C does not have any activity on the HTTP-NG.

Chapter 7

The Ericsson IP Telephony System

Ericsson IP Telephony (IPT) Solution is a XoIP system that offers operators, Internet Service Provider (ISP), and corporate users a standards-based, carrier-class IP Telephony solution that enables phone-phone, fax-fax, PC-phone, and PC-PC services over IP networks. XoIP refers to Anything over IP, for example Voice over IP and Multimedia over IP.

The Ericsson IP Telephony system intends to meet mainstream market requirements for IP phone system ranging from basic arbitrage to more complete enterprise solutions with call control services and user mobility. The system architecture allows existing operators to align and bridge their current investments with the new IP-based communication infrastructure. The system is modeled according to a layered architecture where the layers build on each other and add services to the highest layer in the network.

The Ericsson IP Telephony system is an H.323 based IP Telephony system, but the gatekeeper is split in Sitekeeper and User Agent (see description below) [KHN99]. Furthermore additional services to the H.450 is introduced.

7.1 Architectural issues

The system architecture should identify the key operator roles pertaining to the XoIP system. These are shown in figure 7.1 on the following page. The key roles are:

- Access provider
- Network provider
- Service provider

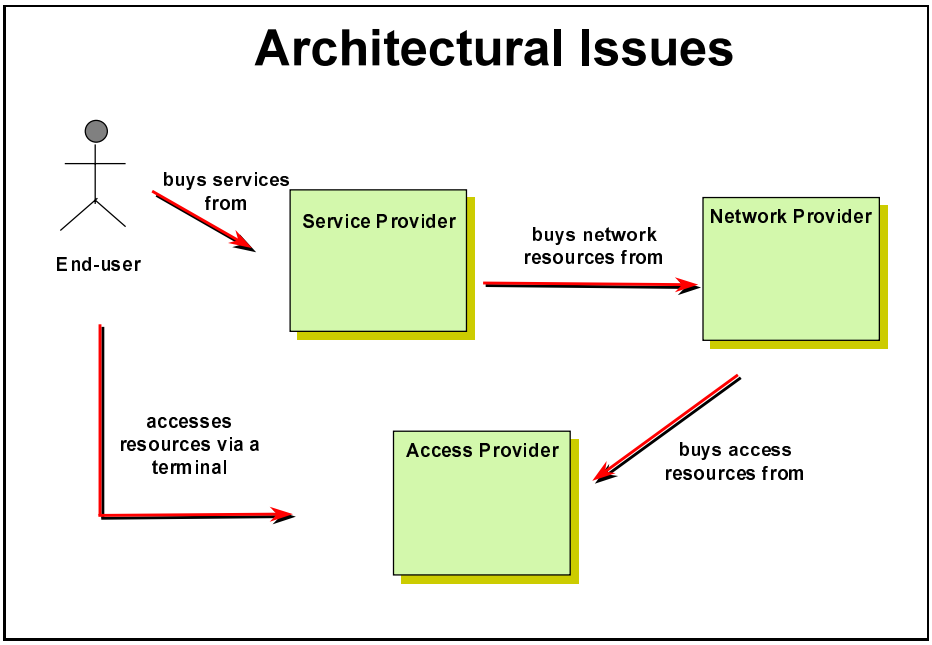


Figure 7.1: Overview of architectural issues from [IPTER]

The end-user buys services from a service provider or a third party service provider, for example through subscriptions. The service provider buys network resources from a network provider, for example, leased line resources. The network provider buys access resources from an access provider, for example through modem-pool or gateway resources in the PSTN network.

The following issues stem from the defined model:

End-users (subscriptions) belong to the domain of the service provider but terminals and end-points generally belong to the access provider. This separation of end-users and end-points means that end-points and end-users need to be modeled as separate managed entities.

The respective providers need to be offered control functions that enable them to control access to the services they offer. For example, the network provider needs to control access to network resources whereas the service provider needs to control access to end-user services. This is; each domain control their own resources. Each network provider does not control access resources directly, but requests an access from an access provider. This access provider has his own policy for controlling his own resources.

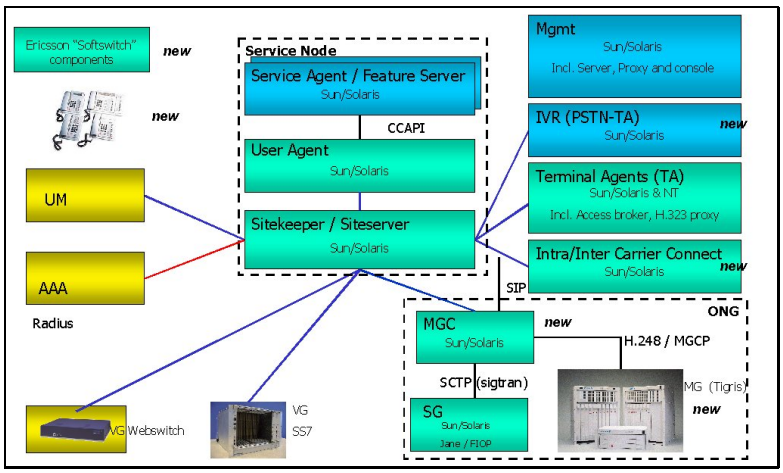


Figure 7.2: Overview of the Ericsson IP Telephony system from [KLI]

7.2 IP Telephony system overview

The IP Telephony system consists of logical nodes. A logical node maps to one or several processes on a target machine. Logical nodes can be co-located on the same physical server. There are several types of logical nodes in the Ericsson IP Telephony system. I will describe some of the more central ones, these are also shown in figure 7.2:

7.2.1 Access Agent

The Access Agent interface an end-user or an external network. An Access Agent that interfaces an end-user is called a Terminal Agent (TA), and an Access Agent that interfaces a network is called a Network Agent (NA).

The H.323 proxy is a Terminal Agent (TA), that is, it interfaces end-users terminals. The H.323 proxy serves two purposes [IPTER]. Additional functions have been added to the later versions of the Terminal Agent. The TA presented in this chapter serves firstly; An end-point behind a corporate firewall needs a proxy to handle both the call signaling protocol, and the media streams. Second; The Service Node (SN) in the Internet Service Provider (ISP) domain needs a proxy for the call signaling protocol. The media streams in only routed via the ISP domain if the endpoint, for example the gateway, is placed in the ISP domain.

The H.323 proxy automatically detects if media is to be routed or forwarded via the H.323 proxy or not. If the H.323 proxy is Terminal Agent (TA) for two connected endpoints, the media is set up directly between two endpoints. If the endpoints are connected to different Terminal Agents, the H.323 proxy will forward the media.

The Access Broker (AB) handles the authentication of H.323 endpoints without H.235 support. There have been no endpoints with support for H.235 security available, therefore the Access Broker must be used when an operator requires users authentication before IP Telephony access. It is possible to configure the IP Telephony application so that user authentication is not required.

The PSTN Voice Gateway (PSTNVGW) is the Gateway between the PSTN and the XoIP network. It acts as an interface between the circuit switched PSTN network and the IP based IP Telephony network. The Voice Gateway takes phone calls from the circuit switched network and puts them on the IP network in a suitable packet format. In the other direction, the Voice Gateway takes packets from the IP network and offers the original stream to the circuit switched PSTN network.

7.2.2 Service Node

The Service node is the logical switch in the system. The Service Node (SN) consists of two logical nodes, the User Agent (UA) and the Service Agent. (SA) The Service node is responsible for IP Telephony services to the end-user. It also handles database issues through the Subscriber and Service Data Repository (SSDR) server who connects to the database. In the rest of this chapter, the term Service node refers to both the User Agent and Service Agent, not these. The term Service Agent only handles the issues considered in the actual Service Agent server.

The User Agent (UA) provides user oriented call services. The User Agent is responsible for the several functions. The User Agent takes care of end-user authentication and registration, H,323 version 2 call handling and control of supplementary service invocation. The User Agent uses the Service Agent for service execution. The interface between the User Agent and the Service Agent is the Call Control API (CCAPI).

The Service Agent (SA) is a logical nodes that is responsible for the providing of end-user supplementary services. The Service Agent and The Subscriber and Service Data Repository will be further described in chapter 8.

7.2.3 Sitekeeper

The Sitekeeper (SK) provides network-oriented services. The Location Server is a logical node within the Sitekeeper. The Logical node provides routing functions.

The Sitekeeper (SK) provides network oriented call services. It handles access to and usage of the network resources within an administrative domain. This includes bandwidth management and non-user oriented access control functions related to location management. The Sitekeeper

also creates data record (CDRs) for all calls.

The Location server (LS) is part of the Sitekeeper. The Location Server provides routing services for the IP Telephony applications. Least cost routing tables are used to find the cheapest route between two destinations based on originating and terminating information. According to [IPTER] Behavior Group Analysis (BGA) is used to modify the A and B numbers and to find the next node (User Agent or Sitekeeper) to process the call. Behavior Group Analysis also finds alternative routes if there is network congestion.

The Ericsson IP Telephony system have decided to split the gatekeeper, so that the Sitekeeper handles the access part of the gatekeeper. The Sitekeeper is related to a physical site and physical VoIP resources [KHN99]. The User Agent, as described above, handles the logical parts, the user and subscriber oriented parts of the gatekeepers functionality.

7.2.4 Management

The Management Node is responsible for the management functions within the system.

The Management node consists of several clients, servers and proxies. The clients Provisioning Client and Management Client contain the Graphical User Interface (GUI). The servers, Provisioning Server and Management Server contain the server functionality. The proxies Management Proxy Client and Management Proxy Server allow management of IP Telephony nodes placed on the costumers LAN from Internet Service provider (ISP) domain.

7.3 The half call model

The Ericsson IP Telephony system is based on a design principle they have called the *half call model*. It is designed to reduce the complexity of a system where several call cases may exists. The key is to model each part of a call as a separate entity or half call. Each half call is then joined together to make a complete call. Figure 7.3 on the following page illustrates the principle.

7.4 Traffic handling

Originating half calls and terminating half calls relates to call cases where PC-to-PC and PC-to-phone, and phone-to-PC is involved, and also calls between phone-phone, and fax-fax. Since I in this paper focus on call cases involving PC, I will only describe these cases.

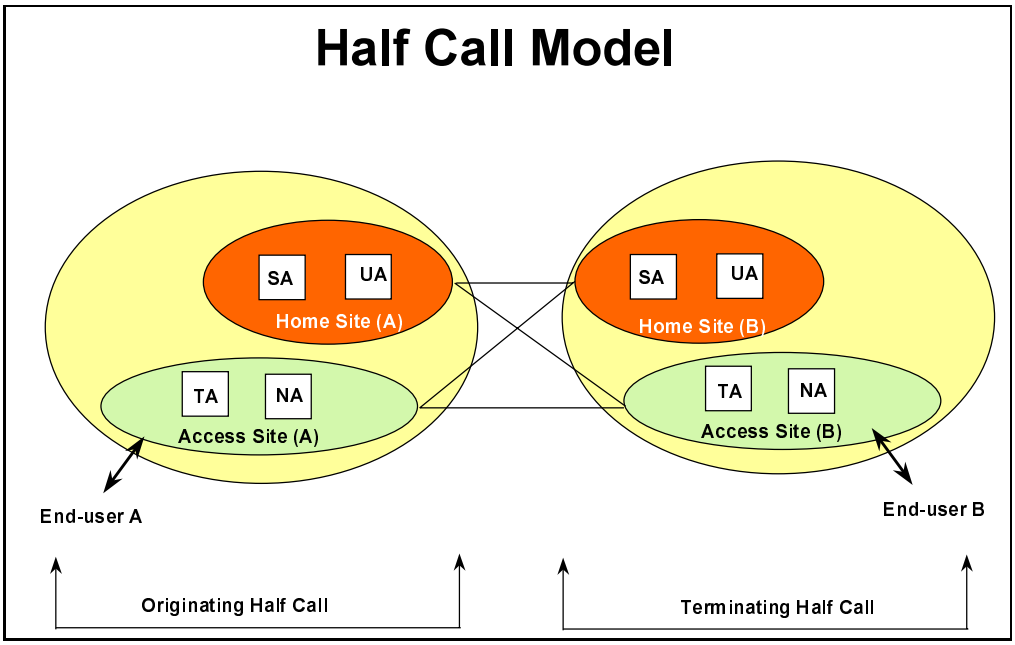


Figure 7.3: Overview of the half call modell from [IPTER]

In an originating half call the signaling procedure sets up a connection between a Terminal Agent connected to the XoIP network on the incoming side. A call from the Terminal Agent always requires a service Node on the incoming side. The Service Node handles originating supplementary services for the calling user. This paper is only discussing a terminating service, and I will not describe the originating half call case further.

In a terminating half call the signaling procedures sets up a connection between a terminal Agent or a Network Access point connected to the XoIP network on the incoming side of a call, and a Terminal Agent connected to the XoIP network on the outgoing side of the call. A call to a terminal Agent always requires a Service Node on the outgoing side. The Service Node handles terminating supplementary services for the called user, as the OnNoAnswer in this case. Before a call can be set up to a Terminal Agent, the terminal Agent must register with a Sitekeeper, the user's terminal must be registers with the terminal Agent and the user must be authenticated by the Service Node.

The call scenario in figure 7.3 relates to PC-to-PC calls, or phone-to-PC calls.

The called party's Service Node receives the setup message. The setup message includes the A-number and the B-number in an international format. The User Agent checks that the terminating end-user is authen-

ticated and that the end-user terminal is registered. The Service Agent executes any terminating supplementary services that may apply for instance the OnNoAnswer, the Personal Agent or Calling Line identity Restriction (CLIR).

The Service Node ensures that the received data does not conflict with the user profile. The Service Node attempts to resolve any conflicts, but is these attempts fail, the call is cleared. The Service Agent now knows where to send the call.

The Sitekeeper then analyses the behavior of the call. The result of the BGA is a type of call and a network accounting level. Three types of routing may apply; least cost routing, UA routing or override routing. The Sitekeeper can now route the call further, and ensures that the access network policy towards the service network does not conflict with the requested QoS.

Upon this the Sitekeeper returns a Call Proceeding message to the Service Node. When receiving such a Call Proceeding, the Service Node sends the Call proceeding to the incoming side of the half call, and the Sitekeeper sends a setup message to the Terminal Agent.

Upon reception of the setup message, the Terminal Agent returns a Call Proceeding message to the Sitekeeper. The terminal Agent then initiates a call to the user terminal. The terminal response is time supervised. If the timer expires, the call is cleared. The setup message is forwarded to the end-users terminal and a Call proceeding message is returned.

An alerting message may be received from the user's terminal without a previous Call Proceeding message. Once Alerting is received from the user's terminal, the terminal Agent sends the alerting to the Sitekeeper. The Sitekeeper sends the alerting to the Service Node, and the Service Node sends the alerting to the incoming side of the call.

Last, a connect message may be received from the user's terminal without a proceeding alert message. Upon reception of the connect message the Terminal Agent sends a Connect message to the Sitekeeper. The Sitekeeper sends a connect message to the Service Node and the Service Node sends a connect message to the incoming side of the call.

7.4.1 User authentication

To subscribe to and to be able to use the IP Telephony system, the end user has to be authenticated in the system. The end-user uses a web-browser to access a web-page that contains the login applet. the applet requires the end-user to enter the user name and password. The end-user host address is read by the applet and included in a HTTP-message over SSL (HTTP-S) to the Access Broker (AB). The Access broker hashes the user name and password and creates the RAS message Gatekeeper Request

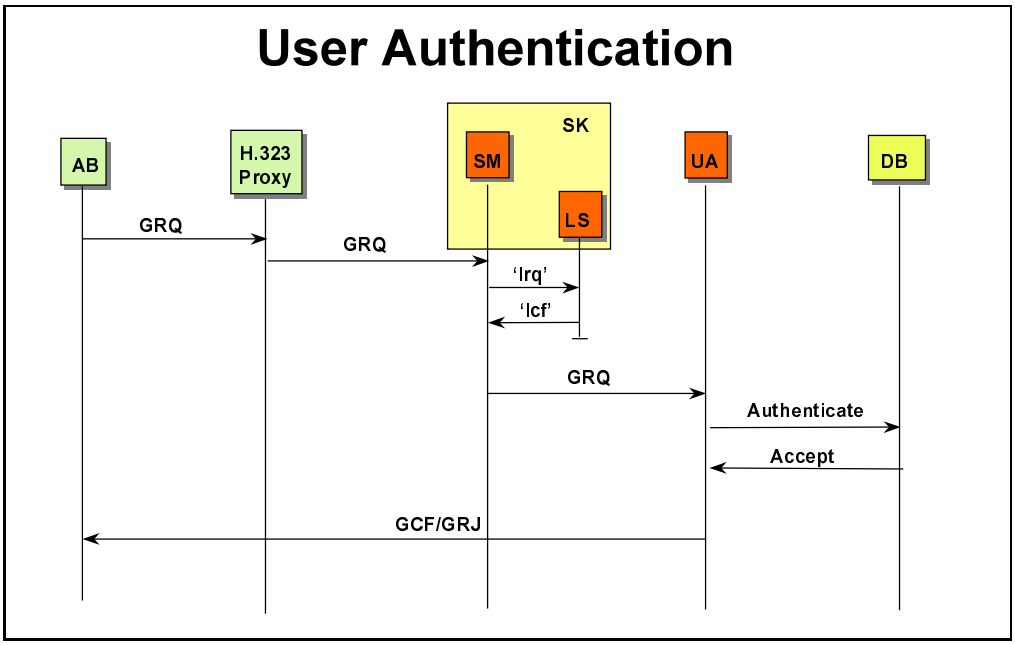


Figure 7.4: Overview of user Authentication from [IPTER]

(GRQ), including in the hashed user name and password, along with the IP address. The IP address is sent so the server knows the path, and to avoid multiple registrations. A GRQ is sent to the H.323 proxy.

The H.323 proxy adds the Access Broker (AB) IP address and its own IP address. performs the proxy functions and forwards the Gatekeeper Request (GRQ) to the Sitekeeper (SK).

The Sitekeeper (SK) comprises the Site Manager (SM) and the Location Server (LS). The Sitekeeper uses the Location Server to determine the User Agent (UA) this end-user belongs to. The Site Manager sends an internal location request (lrq) to the Location Server (LS) and the result, location confirm (lcf) message is returned with the address of the home User Agent (UA).

The Gatekeeper Request (GRQ) is then sent to the home User Agent (UA). The User Agent receives the user name and password and compares them with the stored values.

The RAS message Gatekeeper Confirm (GCF) is returned to the Access Broker (AB) if the user name and password are valid. Access Broker uses HTTP-S to display a message to the end-user.

Should the authentication fail, a Gatekeeper Reject is returned to the end-user. Figure 7.4 gives an overview of the authentication procedure in

the Ericsson IP Telephony system. I will not discussed this issue for the OnNoAnswer service.

7.4.2 Terminal registration

The terminal the end-user uses sends a RAS message Registration Request to the H.323 proxy. The H.323 proxy adds the terminal's host address and the H.323 proxy address, performs the proxy function and forwards the Registration Request (RRQ) to the Sitekeeper (SK).

The Sitekeeper (SK) retrieves the user alias address from the Registration request (RRQ) message and the Server Manager (SM) sends a location request (lrq) to the Location server (LS) to determine the home User Agent for the end-user and returns to the Location server (LS) the home User Agent (UA) for the end-user. Location Server uses network global routing tables to determine the home User Agent (UA) for the end-user and returns the IP address in the location conform (lcf) message. The Sitekeeper routes the Registration Request (RRQ) to the home User Agent (UA).

The home User Agent determines if the user is authenticated and allowed to register. The Service Agent (SA) is notified about a new registration for this end-user. The Service Agent (SA) checks if an additional registration is allowed for this end-user and returns a positive acknowledgment if the terminal is allowed to register. The User Agent (UA) returns Registration Confirm (RCF) to the endpoint.

Should the registration precess fail, a Registration Reject (RRJ) is returned to the terminal.

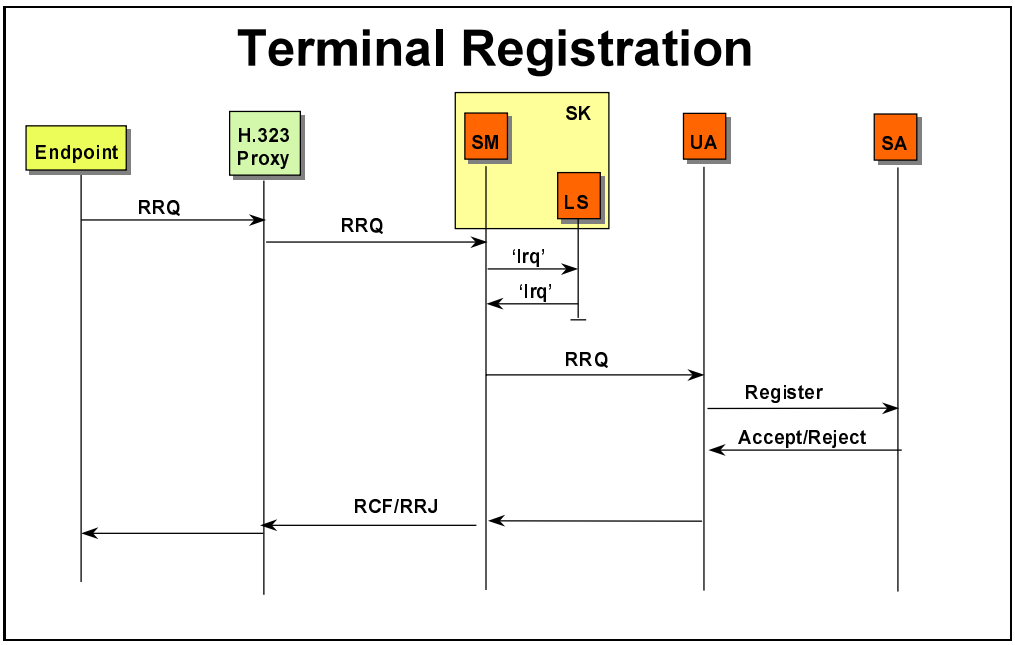


Figure 7.5: Overview of terminal and user registration from [IPTER]

Chapter 8

Designing a Supplementary Service

When designing services for IP Telephony systems, it becomes critical to provide means by which the services to be implemented can be rapidly conceived, developed, and deployed. It should not be necessary to add new network elements for each new service, nor should it be necessary to reinvent the interfaces to existing elements for new services. In addition it should be possible for third parties to create new services easily. [Ros99]

This separation of the servers and the services allows the end users flexibility; they can purchase certain services from one provider, and other services from other providers. As a consequence this also opens new markets for service providers. The Internet, best characterized by its end-to-end architecture, which allows any host to send data to any other host, is seen upon as one, if not the one, ideal platform on which this service model can be achieved.

By third parties, one usually means individuals or organizations beside the ones that own and build the routers, hubs, switches, and servers which actually implements the service (here the Service Agent). This is not the case for the implementation environment for the OnNoAnswer service. This service is being plugged into the server in the core IP Telephony system.
The design and implementation of the supplementary service OnNoAnswer is based in the Service Agent node and closely connected to the Subscriber and Service Data Repository server which is the connection toward the database in the Ericsson IP Telephony system. This location of the service in the actual server implies that the service is being controlled by the operator and not a third party. I will describe some of the functionality of the Service Agent and the Subscriber and Service Data Repository, directly corresponding with the implementation done.

The Service Agent consists of several subsystems: the Service Execu-

tion Environment (SEE), the Service Creation Environment (SCE) and the Service subsystem. Further more it has a CallControll API for communication with the User Agent, an Accounting API for communicating with the AAA-server and connection to a database, the Subscriber and Service Data Repository.

The different services are implemented as plugins in the Service subsystem, read and controlled by the Service Pack. The Service Execution Environment (SEE) is executing the services, and controlling the interfaces towards the User Agent (UA) and the Service and Subscriber Data Repository (SSDR).

The following description is based on [NILS+1999] and [NILS+2000]

8.1 The Service Agent

8.1.1 The Service Creation Environment

By means of the Service Creation Environment a service developer can create a service pack with a set of service functions that can be plugged into the SEE. The Service Pack designates the logic responsible for instantiation of the service logic (service plug-ins), sequencing of the different parts of the service logic and interactions between the service logic parts. The Service Plug-in constitutes the actual service logic for one specific supplementary service. Each service plug-in shall preferably be a small-dedicated class handling a well-defined set of tasks. Example wise the service logic needed for out-going call barring might be implemented as a service plug-in. An overview of the Service Agent is presented in figure 8.1 on the next page

8.1.2 The Service Execution Environment

The IP Telephony system provides a Service Execution Environment allowing supplementary services to be executed. The SEE is connected to the User Agent via the Call Control API (CCAPI). The SEE is built on the Java server framework and is managed via the management system.

The SEE interfaces the User Agent via the Call Control API, and the Subscriber and Service Data Repository. The CCAPI connection to the User Agent is used to register User Agents in the SEE, control user registrations in the User Agent and control the call handling in the User Agent.

The useragent subsystem in the SEE is responsible for defining user agents allowed to connect to the SEE, receiving notifications about user agent registrations and keeping track of the listed users agents connection status. The maintenance of allowed user agents is performed via the management system interface.

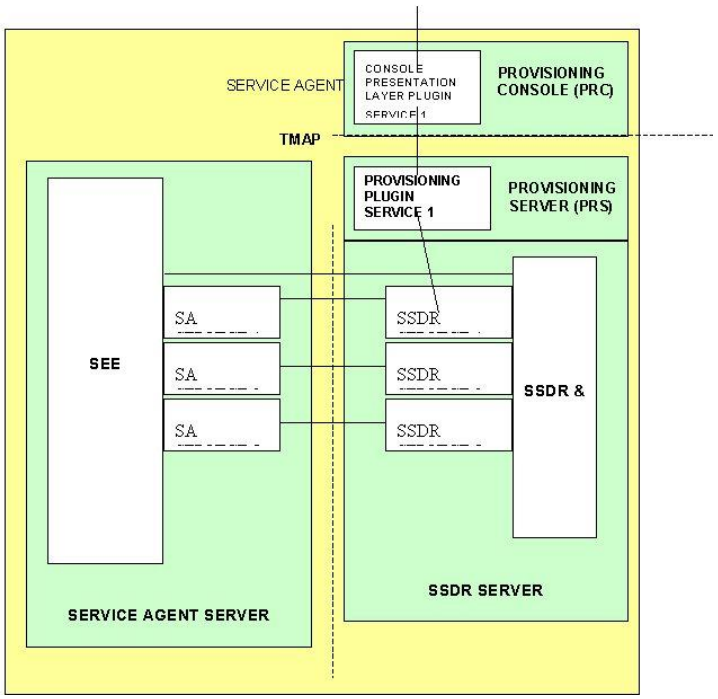


Figure 8.1: Overview of the Service Agent from [NILS+1999]

The user subsystem in the SEE is responsible for handling user registrations. Registration attempts towards the user agent is notified to the SEE. The SEE will notify the servicedata subsystem and transfer control to a registered user registration plug-in. The plug-in is then allowed to accept or reject the user registration. De-registrations towards the user agent is also notified to the SEE and the user subsystem which will update the user status and notify the servicedata subsystem. The registration status for users in the system is maintained by the user subsystem, but is actually contained in the user status repository in the CCAPI.

The servicedata subsystem provides a cache where service data is stored and retrieved. The servicedata subsystem allows service data cache handlers to be registered for maintenance of the service data cache. A service data cache handler will register it self in the user subsystem to be notified on registrations and de-registrations. Typically a service data cache handler will fetch service data from a data store upon the first registration of a user and revoke the data when a user cease to exist in the SEE. The SEE provides on service data cache handler which is responsible for maintaining the service data cache with respect to the SSDR. The service data cache itself provides functions used by the services to access their service data.

The Subscriber and Service Data Repository subsystem is responsible

for interfacing the SSDR. The subsystem provides functions used by the SSDR service data cache handler to fetch service data from the SSDR. The subsystem also provides mechanisms for re-reading of service data upon notifications from the SSDR that the service data is changed. The session-control subsystem is responsible for creating service sessions upon receipt of new service set-ups from the CCAPI. A session factory is registered with the subsystem and is called to create new service sessions. The session factory will typically instantiate a service pack which is returned to the CCAPI and later called to handle the events in a service session.

8.1.3 Call Control API

The Call Control API is a Java based interface to the User Agent. The API provides mechanisms for notification and control of User Agent registrations, User registrations and Calls. The API allows an application to control the User Agents behavior. The underlying protocol in the API is network (TCP/IP) based and allows physical distribution between the controlling application and the User Agent. The typical usage of the API would be to provide value added service on top of the User Agent, such as number translation, screening etc. For the remainder of this document the application using the Call Control API will be denoted as the CCAPI client, where CCAPI is short for Call Control API.

The CCAPI client can receive events from the User Agent and issue action requests towards the User Agent, this is called CCAPI/User Agent interaction. CCAPI/User Agent interaction is divided into connection related and the session related interaction. The latter deals with everything during call lifecycle, the former with everything else.

8.1.4 Session related interaction

The session related interaction is basically what goes on between the User Agent and the CCAPI client during the call lifecycle and resembles the call model of the User Agent. The CCAPI client side of a call is titled a service session in this document. Generally all session related interaction will be handled by an event observer in the CCAPI for each session (CallControllerEventObserver). Session related event is one of two types, either a subsequent action request is allowed and required or not allowed and not required. The first is events reported by monitors (event types) that are interceptable, the latter is events reported by monitors that are transparent.

CCAPI /User Agent interaction is based on a protocol where the events from the User Agent and the action requests from the CCAPI client is transported via Java serialization mechanism. A TCP connection is set up between the peers and a pair of object streams (input and output) is created on the socket connection to carry the serialized objects. The CCAPI will translate to and from the serialized object protocol and offer a straight

interface to the CCAPI client with notification interfaces for reporting of events from the User Agent and interfaces implemented by the CCAPI where the CCAPI client can issue action requests.

8.1.5 Connection related interaction

The connection related interaction involves User Agent registration and de-registration, user registration and de-registration and connection monitoring. In addition the initiation of a session (SelectInvokeOrg/Term) is done via a connection related event. Generally all connection related interaction will be handled by an event observer in the CCAPI for each allowed user agent (UserAgentEventObserver). All connection related events from the User Agent requires an acknowledge or reject from the CCAPI client. The only exception is the initiation of a session that is either rejected directly or later confirmed with an action request from the allocated session. The only action request that can be issued related to connections is a user de-registration, this action request will be acknowledged or rejected by the User Agent.

8.1.6 Supplementary Services

A supplementary service consists of four implementation components:

- Service Execution Environment (SEE) Service Plugin
- Subscriber and Service Data Repository handling
- Provisioning Server (PRS) Service Plugin
- Provisioning Console (PRC) Service plugin.

The subsystem components are included in different Java applications (servers) running in separate JVMs. The SA service plugins execute in the Service Agent Server as plugins in the Service Execution Environment (SEE). The SSDR service plugins execute in the SSDR Server as plugins in the SSDR framework. The PRS service plugins execute in the Provisioning Server as plugins in the Provisioning framework. The PRC service plugins execute in the Provisioning Console application as plugins in the Provisioning Console framework.

8.2 The Subscriber and Service Data Repository

The Subscriber and Service Data Repository is the server connecting to the database in the Ericsson IP Telephony system. The nodes in the IPT system use persistent data for various purposes. The Subscriber and Service Data Repository (SSDR) forms a common storage for such

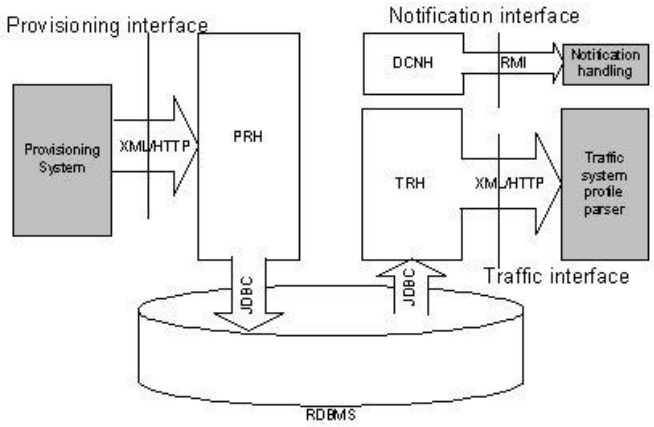


Figure 8.2: Overview of the Subscriber and Service Data Repository from [NILS+2000]

persistent data. The SSDR is characterized by being based on a relational database system, having an open interface towards the traffic nodes for data retrieval based on HTTP and XML and by having an open interface towards the provisioning system for data insertion, update and retrieval based on HTTP and XML. Figure 8.2 gives an overview of the Subscriber ad Service Data Repository.

In customer installations of the IP Telephony system the operators have their Ericsson IP Telephony system and IP Telephony as one of many service offerings to their customers. Hence the implementation needed by the IP Telephony system is normally a subset of the complete information about the operators customers.

The Subscriber and Service Data Repository shall handle the following entity classes:

- Subscribers. The subscriber entity corresponds to a user of the IPT system. The subscriber is assigned a set of attributes that is used by one or many traffic nodes.
- Services. The service entity corresponds to a service being offered by the IPT system to its subscribers. A service may have various attributes being used during service execution by the IPT traffic nodes. In the SSDR each type of service being provided by the IPT system is regarded as a separate service entity.
- Service subscriptions. The service subscription entity is used to form the relation between a service entity and one or many subscribers subscribing to the service. One subscriber may only have one service subscription to each service type.

A third party system shall be able create a subscriber and/or service data entry, update its content, get the content and remove it.

8.3 Technical outline

The Subscriber and Service Data Repository comprises three main parts. Each part has a well-defined interface towards their clients.

- Provisioning Request Handler The Provisioning Request Handler (PRH) maintains the data contained in the SSDR. The PRH provides an interface towards 3rd party or IPT clients that manages the data in the system. This interface is denoted the Provisioning Interface. The PRH will perform date insert, update, delete and retrieve actions according to requests received on the provisioning interface.
- Traffic Request Handler The Traffic Request Handler (TRH) feeds the traffic nodes with data from the SSDR. The TRH provides an interface towards the traffic nodes that uses the data in the system. This interface is denoted the Traffic Interface. The TRH will retrieve data and deliver to the traffic nodes according to requests received on the traffic interface. The TRH will also receive notification subscriptions and desubscriptions from the traffic nodes. However the actual handling of change notifications is performed by the DCNH.
- Data Change Notification Handler The Data Change Notification Handler (DCNH) notifies the traffic nodes when data in the SSDR is changed. The DCNH provides an interface towards the traffic nodes wanting notifications upon changes in the SSDR data.

The provisioning and traffic interfaces to the SSDR are based on HTTP. This is favorable in the sense that it is easy to understand, easy to test and it is supported by most major operating system and programming languages. One prerequisite for using HTTP is support for HTTP1/1 allowing persistent connections. The GET method is used to perform requests for content from the SSDR, e.g. for subscriber data. The POST method is used to submit new content to the SSDR, e.g. posting a new data for a new subscriber or for submitting updates. The change notifications are delivered by means of RMI.

Chapter 9

Some Existing Services

9.1 Traditionally supplementary services

Using the ITU-T definitions [TAHN] telecommunication services can be divided into bearer services, telecommunication services and supplementary services. Today almost all communication devices have the # and the * button and the introduction of new communication services are growing rapidly. In addition GSM phones have display, yes and no buttons and text possibilities.

Traditional supplementary services are all standardized and tightly coupled with the call signaling, for example the several call forwarding services are built into ISUP. This standardization has a few implications.

- The need for agreement between operators does not support competition.
- The standardization procedures is time consuming.
- Introducing a new service implies the need to update all switched along the call signaling.

I will in this section describe a selection of the most common used traditionally services offered. The following services is described to give an introduction of the basic from which the idea on implementation of the OnNoAnswer service is being generated. Other services exists as well, such as call waiting, outgoing call baring etc.

9.1.1 PSTN/ISDN supplementary services

I will shortly describe some of the call forwarding services in an PSTN/ISDN network. The call forwarding services makes it possible to transfer calls from your home phone to any other terminal, that is mobile phone or other subscribers mobile or home phone. It is possible to active this service

both from your own home phone or from other terminals. The latter is titled "Remote forwarding".
The call will first be directed towards the called party's end-point, but instead of alerting on the end-point, the service part of the local exchange (see figure 2.1 on page 12 in chapter 2) and will transfer the call to the preferred forwarding terminal. There are several different forwarding options available today:

- Call Forwarding no reply is an option that works in the same way as described above. When the subscriber do not answer the call after 20 seconds of signaling the call is transferred to another preferred terminal.
- Call Forwarding when busy in another call works similarly. The caller will be transferred to another preferred terminal when the called party do not answer on alerting.
- Call forwarding unconditional is an option that forwards the calls directly to another preferred terminal, without alerting on the subscribers home phone. That is the call is never routed to the subscribers home phone, but are redirected to another terminal in the network.

9.1.2 GSM supplementary services

In GSM the most basic telecommunication services supported is telephony. Also a variety of data services is offered. GSM users can send and receive data to users on PSTN, ISDN, Packet Switched Public Data network, and Circuit Switched Data network using a variety of access methods and protocols. The basic services, as several call forwarding services as described above, is also implemented in GSM.

Supplementary services are provided on top of telecommunication services or bearer services. In the Phase 1 specification of GSM, they include several forms of call forward as unconditional, on no reply, on busy and on not reachable, call waiting, follow me and call barring of outgoing and incoming calls, for example when roaming in another country. In Phase 2 of GSM, specification caller identification, call waiting, and multi-party conversations were introduced. The way to standardize services is a great disadvantage of the existing GSM network [SKO]. New services cannot be implemented without an extensive standardization work.

9.2 IN

The basis of Intelligent Network (IN) is that the services and the network is . In GSM this is done by means of the CAMEL architecture.

Intelligent services are services based on the IN architecture, these

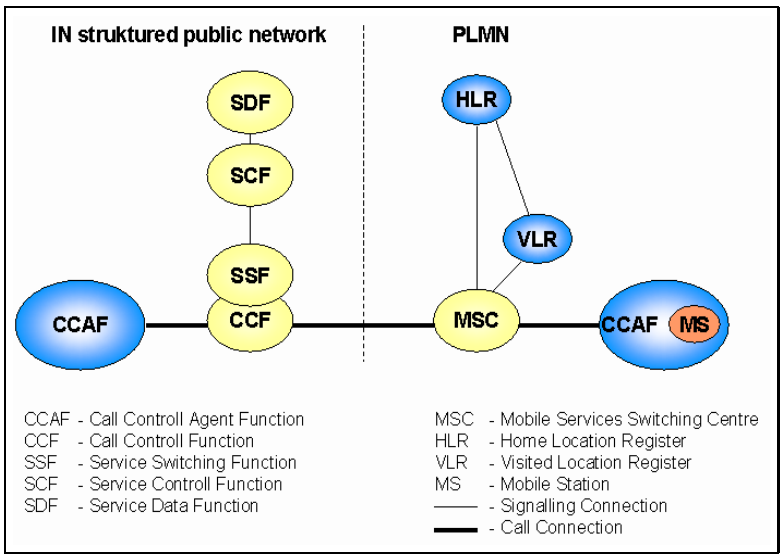


Figure 9.1: Overview of IN architectural issues from [SKO]

services can by classified as value-adding services in the content that they add some kind of value to the actual user of the service. These value-added services can be realized through the means of the IN architecture or external computing systems.

9.2.1 IN architecturefor PSTN/ISDN

Intelligent Networks (IN) are networks which has differentiated services and signaling. This concept of separating the service logic from the server, which are the fundamental of developing and deploying services in an IP telephony network, is also the heart of IN, which today is a key component of the telephone network. The In arose out of the need for separating services from the telephone switches, enabling rapid development of new services. [Ros99]

IN is a collection of standards codified by the International telecommunications Union, telecommunication Section (ITU-T). The architectural idea of IN is modeled as a number of distinct planes. [SKO] has illustrated an IN structured public network like this:

Starting from the top, the first layer is the service layer, which contains services, such as call forwarding or free-phone. The nest layer defines these services in terms of atomic operations, called Service-Independent Building Blocks [Ros99].

[TAHN] describes an Intelligent Network (IN) in terms of functional entities and the physical entities that performs these functions. The IN functional entities for ISDN/PSTN are shown in the left side of figure 9.1.

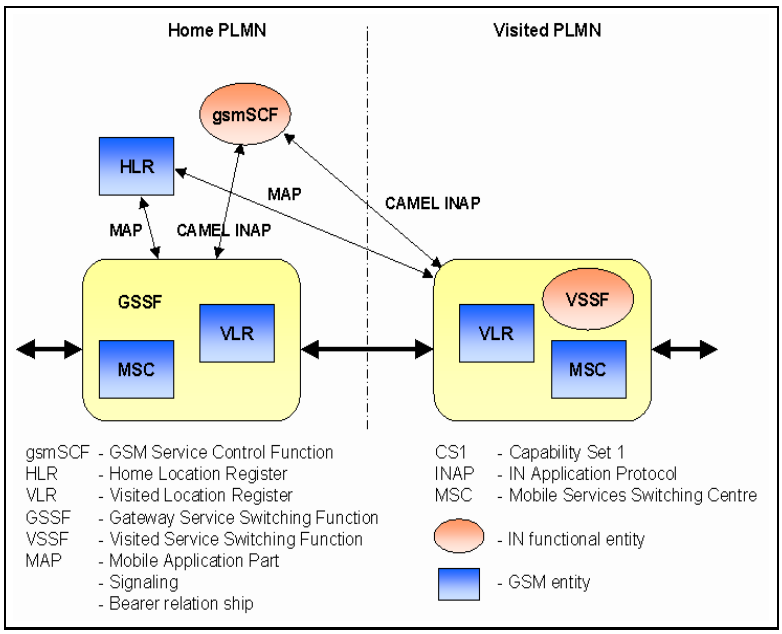


Figure 9.2: Overview of the CAMEL architecture from [SKO]

The call control agent function (CCAF) provides end-user access to call and service processing. The service switching functions and call control function (SSF/CCF) provides basic switching functions and calls control, and can detect events that require invocation of IN server logic. The SSF allows the CCF to be controlled by the SCF. The CCF handles non-IN call and connection control. The CCF and SSF are typically co-located and may communicate via a proprietary protocol. The service control function (SCF) contains the IN service logic required to handle IN provided services and provides and provides service execution control. It is notified by the SSF when certain events occur, such as when certain numbers are dialed, for example an UPT number, and can request the SSF/CCF to perform functions needed to provide services and billing. The service data function (SDF) contains the data necessary for execution of an IN service and may perform data processing functions. As such, it provides a logical data management function for SCF, including handling access and providing consistency checks. The service management function (SMF) contains the entities dedicated to the management of service, according to the IN architecture and devices, which can be used by other network entities, for example synthesized voice announcements. The SRF interacts with users via connections to play announcements, collect digits, and, in the future, translate text to speech, perform voice recognition etc [TAHN].

9.2.2 IN architecture for GSM

Due to the limitation of introducing services in GSM mostly cased by the exhaustive standardization work a new feature called CAMEL was developed [SKO]. The Camel was meant to be a feature to integrate Intelligent network and GSM, and is a toolbox for operator specific services, with the possibility for tailor made services. This means that the Camel features may be used by network operators to provide customized services to mobile subscribers also when roaming outside the Home Public Land Mobile Network. Thus CAMEL is to a large extent based on IN for fixed networks, with som changes. The CAMEL architecture is illustrated in figure 9.2 on the preceding page.

9.2.3 IN services in PSTN/ISDN

There are several services based in an IN architecture. The most common used In services are free-phone, universal access number (for example 815-numbers) and voting services.

I will in this section describe the Universal Personal Telecommunication (UPT) as an IN service, looking beyond the fact that this is not actually a service being offered today. Nevertheless, the ideas behind UPT is important, and offers an important aspect of personal mobility.

Norwegian Telecom operated a UPT pilot service the autumn of 1994. The study was ended spring 1996, due to lack of success of the UPT function [TAHN]. According to [TAHN] the failure of UPT where mostly the cumbersome ways for the user to identify himself to the network. Today it is possible to have a key, based on tones, that could identify the user, so that the tiresome and cumbersome identification process where minimalised.

The idea behind the UPT is the identification of the trend that users of communicating service are growingly more mobile. A user wants to be accessible to keep close contact with their supporting base, both private and business. Since the OnNoAnswer service discussed in this paper is looking into the personal user aspect of a service I will concentrate on this also in the description of the UPT concept. Users that move around according to a fairly stable time schedule, for instance between work and home, may for convenience reasons want to be able to receive calls as he or she moves from place to place without making any active action.

According to the definition of UPT, UPT gives the user the possibility to access telecommunications services from a variety of terminals and network access points from a number of networks. The terminal the user is using for access may be attached to a PSTN/ISDN network, a GSM network or an IP-based network. The geographical location of the terminal

the UPT user wishes to use for his communication needs is of no concern for the UPT service. As such UPT offers personal mobility for the users with regard to both terminals and networks. However, there may be some restrictions imposed on the UPT service by the capabilities of the networks and terminals.

This represented a new way of thinking about personal mobility for users of telecommunication services. Telecommunication services where mostly network-centric, and the UPT service gave the users to access the network, and by the means of this the possibility to access their services from any network including their home network.

The principal characteristics of UPT is the personal mobility, meaning that UPT users will be able to move between terminals on a global basis in order to access telecommunication services at any terminal and making and receiving calls on the basic of a personal identifier, and the capability of the network to provide those services according to the user's service profile [TAHN].

Through UPT telecommunications services is provided on the basis of the identity of persons rather than the identity of network access points, as for example in the traditional connection between network access points and identifier in fixed network. All UPT users will be allocated and associated with their own personal UPT number, which acts as a basic personal user identification. This UPT number is network transparent. The UPT number is public and is dialed by others when they want to contact the UPT user. The main purpose of this is for the UPT user to be identifies with only one number, instead of a E.164 number, a cellular number for his mobile, and an e-mail address.

The network operator or service provider also identifies each UPT user by another personal user identity for security and management purpose. It is unique to each subscription and all information that is unique for a subscription is related to this number. Still, UPT user will know the number this identifier is not public,, as it will have to be supplied to the network every time the UPT service is accessed.

As it where, to access the service the UPT user had to dial a general service access number whish is used when the UPT user wants to make an outgoing call or make an registration or check the status of his/her present registration or service profile. This dialing turned out to be to cumbersome, but with today's means there is other option for a user to uniquely identify himself, this could be done by automatic key-tones or a smart-card, for example.

The UPT service provider will in principle offer access to a range of available telecommunication services from which the UPT subscriber selects a set. As such, UPT is a service system concept that acts as an umbrella for a number of telecommunication services. The UPT subscription may typically includes several options such as basic telecommunication

services and supplementary service [TAHN].

9.2.4 IN services in GSM

Telenor InTouch is a rather new forwarding service that is developed based on Intelligent Network (IN) technology. The core of the service is that the user can by means of his or hers cellular phone decide when and on what terminal the users wants to answer a call. This makes it possible for the end-user to control his or hers availability. The service is available only for those who subscribe on a home phone and a GSM mobile phone from Telenor.

TouchIn is a service the end-user subscribe to. The end-user registers, and operators will connect the end-user to the service. By default, the calls are forwarded from the subscribers home-phone to his cellular phone. The subscriber have a limited options for forwarding the call. It is possible to forward the call to the following:

- The Mobile answering service if the call it not answered
- The mobile phone if the call is not answered. This is the default setting.
- Directly to the mobile phone if the terminal is on.
- Direct to another number from your home phone

The subscriber can alter the settings by means of his cellular phone, the Internet or WAP. This makes it possible for the user to change his prefer terminal on the fly.

The user can change the settings by the cellular phone or his home phone by calling 05555. He can also change the settings by logging onto the InTouch home page, register with a pin-code and InTouch number. It is also possible for the user to alter the settings by WAP. Furthermore some option exists to route calls differently based on the callers identity.

9.3 IP Telephony services

There are today several services that is offer to the end user in an IP Telephony system. Some of these services are offering the end-user some of the functionality the OnNoAnswer service are offering. I will therefore describe a few of these services to highlight some features that might be important when considering the functionality, and use, of the OnNoAnswer service.

The IPT Telephony system provides the end-user some supplementary services and applications. These are Personal Agent (PA), Virtual Private Network (VPN), Private Numbering Plan (PNP), Outgoing Call

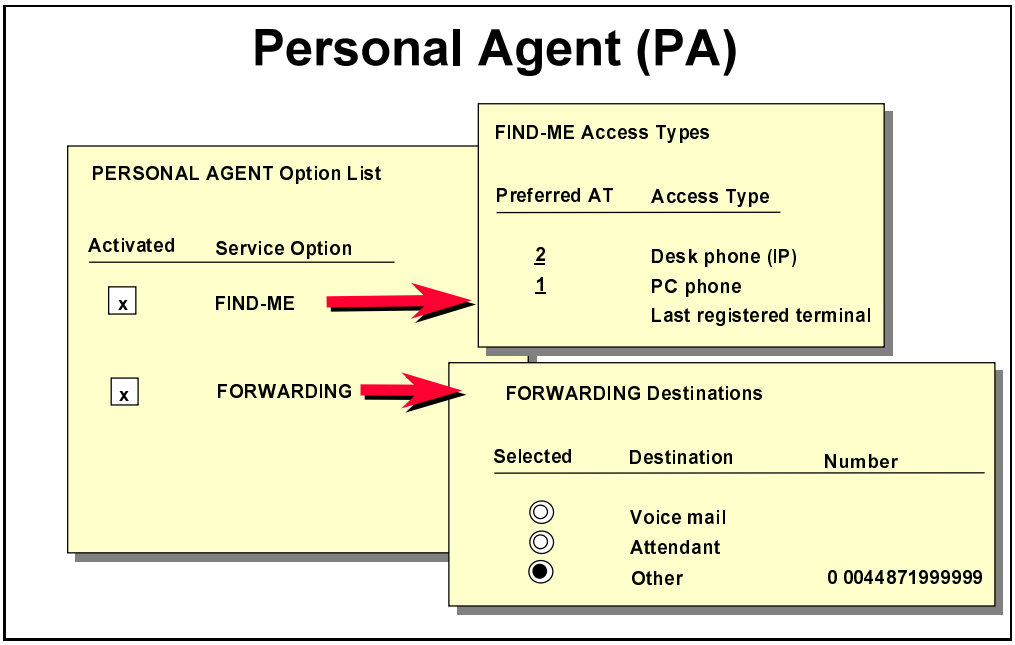


Figure 9.3: Overview of the Personal Agent service from [IPTER]

Barring (OCB), Calling Line Identification Presentation/Calling Line identification restriction (CLIP/CLIR), Call restriction (CR), and Calling Card (CC). I will illustrate the Personal Agent service.

9.3.1 The Personal Agent services

The PA service is a terminating service that allows the service provide configure how terminating calls to and end-user is handled. The service also defines what type of terminal and how many terminals can be registered simultaneously by one user.

The PA service provides the capability to offer incoming calls to predefined accesses types in preferred order (find-me-option) or forward the call to a specified destination (forwarding option). In line with the user's service profile, incoming calls to the user are delivered to the terminal of preferred access type, if registered. If the terminal is not registered, or if the call setup is not successful, the call is directed to another type currently registered, before proceeding with the forwarding option.

The PA service ensures that by dialing one number, the calling user can reach the called user regardless of where the called user is currently registered.

A terminal is considered registered in the system when it has been swished on by the end-user and authenticated by the system. Each

end-user subscribing to the PA service can be registered at two terminals of different access types simultaneously.

The PA service has two options for terminating calls to an end-user:

- The find-me option allows the user to be registered with up to two access types, currently a PC-phone and a desktop phone. The operator and the end-user configures the priority used when terminating a call to the end-user. the system attempts to complete the call to each access type in turn. If the system fails to complete the call using the find-me option, the system can be configured to use the forwarding option to complete the call.
- The forwarding option allows the operator and the end-user to configure how to forward a call that is not terminated by the B-number or the find-me option. The forwarding option allows the operator to configure up to three different numbers where only one of the numbers is active at any time. The numbers are typically an operator attendant, a voice mailbox or another forwarding number.

End-user management of the Personal Agent service

The end-user can access the service profile through a web-interface specified by the service provider. The end-user controls the execution of the PA service by maintaining the following:

- Which service option are active
- The preferred access type from the available access types specified by the provisioning administrator
- The forwarding destination from the available destination specified by the provisioning administrator.

Management of the Personal Agent Service

The provisioning administrator can control which default service features are provided. The provisioning administrator can also control which access types and destinations are available for the service features.

For the find-me option, the provisioning administrator may on behalf of the end-user, select a preferred access type. If no preferences are specified the system selects the last registered terminal as the preferred terminal.

For the forwarding option, a default destination type can be set by the provisioning administrator but is not mandatory. If specified, a number must be entered.

Provisioning and maintenance of the Personal Agent service profiles



Figure 9.4: iPulse display with menus and buttons from [IWP]

at an individual level can be cumbersome and time consuming. To address this, the provisioning administration can provide a group of users with a default profile. Find me access types and forwarding destination lists can then be defined or changed at group level. Onthe other hand; having the possibility of individual service profiles for the end-user, that is by means of having the “other” option in forwarding configured by the end-user.

9.4 Convergence and the future

9.4.1 iPulse

The description below is based on [IWP]. This information is also availiable from [IER]. iPulse is an existing service. It is a is a communication portal that connect users by computer, PDA, or mobile phone through a point-and-click contact menu offering value-added services. It is a communication technology that links the Internet with wireless and wire-line telephony networks.

The iPulse framework acts as a communication broker [IWP]. It handles communication between two or more parties. The contact list, buddy list, is stored at the iPulse server, and not on any terminal device. This network-centric design makes it possible to have the same services on the PC, the mobile phone or home phone, that is on any terminal at any location. This also integrates wireless and wireline telecommunication

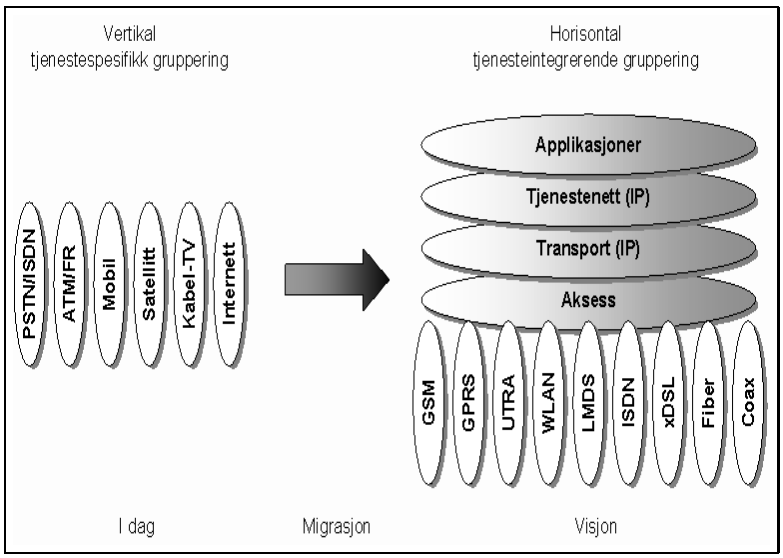


Figure 9.5: Overview of service architectural issues from [TAHN]

with IP communication [IWP]

One of iPulse features is that they offer control to the user over who the end-user can be contacted by at given time and device. The upcoming iPulse 1.5 release intends to offer functionality that gives the user control over all communications and added privacy. The feature offers the user to customize the communications by setting up individual profiles that indicate when, by whom and how the user want to be reached. This is done by iPulse routing the communications you will receive to the terminal you have asked for the communication to be placed at.

You can specify your own status that will be displayed to other users. This will allow other users to determine the best way to reach you. In OnNoAnswer it is the subscriber to the service that decides what is the best way to be reached, that is B, not A. A can think it's important, but not B. Who shall decide on intrusion?

But, it is possible for the subscriber to customize various aspects of the system, such as your personal information, as well as communication preferences, such as how you preferred to be reached.

The subscriber on the iPulse service can edit and add contact list entries and groups. The example shows colleagues, friends and family.

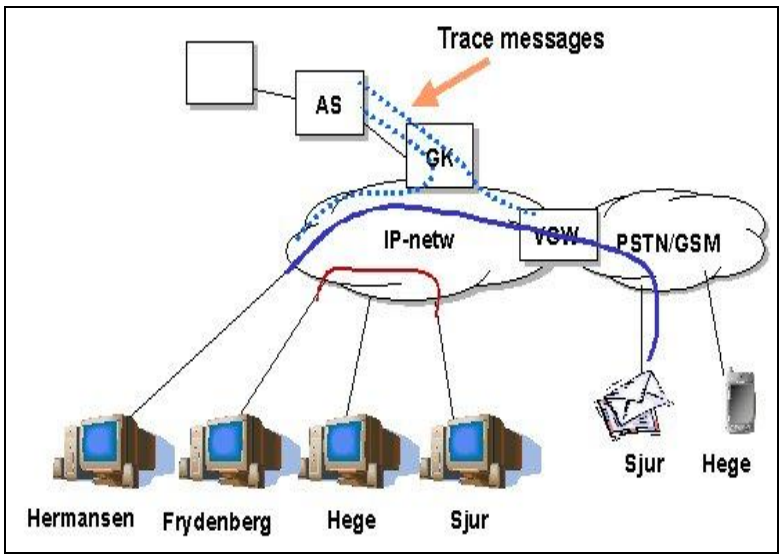


Figure 9.6: Call flow using OSA [PRO]

9.5 Migration

Figure 9.5 on the preceding page gives an overview of how services are deployed today. [TAHN] suggest that a migration will happen, and that different network will access a common access API to access services. This indicates that services will no longer be associated with the networks ability as it is today, but merely to the services subscribed to. Limitations to terminals and communication devices will although restrict the use of different services being subscribed to, and media negotiation will be required.

9.6 Some service prototyping using OSA

A cooperation between Telenor and Ericsson took place winter 2000. The goal of the project was to prototype according to the Open Service Architecture. The OSA is an architecture that will ensure open interfaces for third party application developers as well as open and secure interfaces towards service and content providers [DIP]

Figure 9.6 illustrates the call flow of an call forwarding to voice mail on busy using OSA. The OnNoAnswer service adds a new scenario to this illustration. As the figure shows there are noe interaction between the caller and the called party. The call flow is one directional, that is the gatekeeper detects that the called party is engaged in a call, and transfers the call directly to voice mail. By adding means for supporting interaction exchange messaging, the OnNoAnswer service response options could be posted back to the caller.

Chapter 10

Implementing the Service

The Ericsson IP Telephony system provides the basis for the implementation done in this thesis. The OnNoAnswer service is developed by means of the Service Creation Environment described in chapter 8.

By making the service as a plugin, it could easily be employed in the Ericsson IP Telephony system. During prestudy and design it became clear that the service could not be implemented directly into the system. Since the current version of H.323 standard did not support identifying and sending URLs during call setup messaging, it was not possible to implement the service as a valid service. Since the service could not possible run in the environment based on the standard as it is today and due to safety reasons, management at Ericsson decided that the service should be developed and run in a separate environment, using copies of parts of the system directly referring to the service.

As described earlier the function of the OnNoAnswer service is based on annex K as it is outlined in the various standardization papers and draft [ITUTD-13]. This proposal and description is meant to be a general one for the H.323 standard, and by means of that, possible to employ and support by all H.323 compatible networks and end points.

More precisely a copy of the Service Agent node in the Ericsson IP Telephony system was used for running the service. A User Agent simulator simulated the call setups, with parameter for terminating and no answer conditions. A Jigsaw web-server was used for interacting over the WWW-interface. The programming language was Java and the development and implementation environment was Borland JBuilder 3.0. Due to safety reasons the implementation took place at Ericsson at Billingstad.

10.1 The Service Architecture

When the service is called in on the right conditions in a call, the plugin should identify the calling party's number and make a lookup in a database. Based on this information the database should return a URL, containing the relevant information and options for the calling party identity. This URL should then be loaded back to the calling party's browser, and the calling party should be able to interact with the further pressing of the call.

The conditions that the OnNoAnswer service should be called in on could be one of the following:

- On alerting for a pre-defined time
- On busy
- Always

Another aspect of the service is that the subscriber should be able to maintain the service configurations by interacting with a web interface. By calling himself, that is when the calling number is identified in the database as the subscribers identification alias, an URL containing contact-lists and information-strings should be loaded to the subscribers browser. The subscriber can then modify and update the information.

This means that when the user calls himself, he is automatic routed to the configuration interface. This is much the same as in todays GSM voicemail. This does however mean that the user cannot call himself; normally one never needs to, except for testing purposes anyway. The relevant information located in an URL returned to the calling party, if the calling party is not the subscriber, should be:

- Options for how to proceed the call.
- A block of information telling the calling party why the call is not being answered. This block of information is a selection of many based on the calling party's identity.

The relevant information located in an URL returned to the calling party, if the calling party is the subscriber, should be:

- Editable lists over all categories of contacts.
- Editable information strings.
- Editable choices for how to proceed the call.
- Possibility for setting the preconditions for when the OnNoAnswer service should be activated.

When describing the work done with implementing the prototype I will separate the OnNoAnswer service into four main parts:

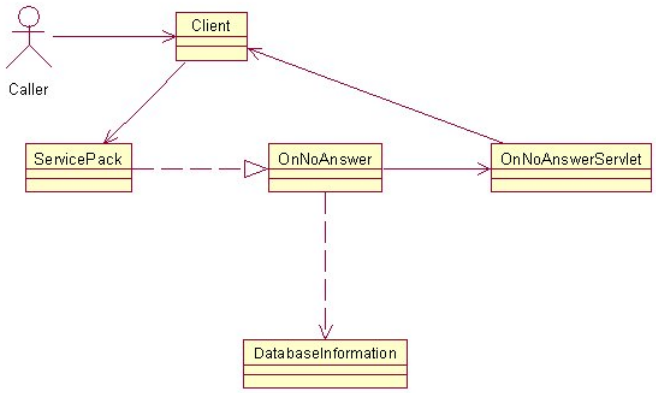


Figure 10.1: Class digram for the OnNoAnswer service

- The resolution of execution order when the call is initiated
- The insert, update or retrieve of information from the database, the Subscriber and Service Data Repository (SSDR).
- The posting of the service back to the calling party
- The further processing of the call

These are the main parts of the service that are implemented. How the user enters the IP Telephone system is for most part described in the previous chapters, with weight both on general IP Telephony solutions and the specific Ericsson IP Telephony solution.

Furthermore, when designing the service four different modules where first identified:

- The Service Pack
- The Service Plugin
- The service Servlet
- The Database Information

This is also described in figure 10.1.

Figure 10.1 shows the components of the OnNoAnswer service. The caller interacts with the IP telephony system by means of a HTTP-client, a web-browser, in this prototype. This client initiates a call. The call is entering the IP Telephony system, and enters the Service Agent from the User Agent via the CallControll API. This is not shown in the figure.

When the Service Agent is being called in on the terminating side of

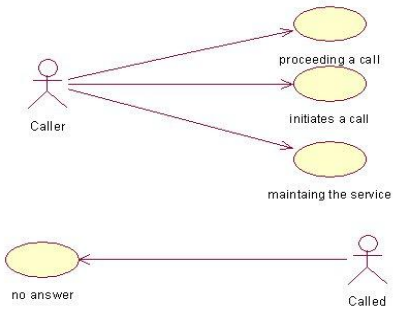


Figure 10.2: Central use-cases for the actors

the call, the Service Pack is resolving the execution order, and the OnNoAnswer plugin is being called in, instantiating a session on behalf of the subscriber. The OnNoAnswer plugin is execution a query to the Subscriber and Service Data Repository, based on the alias of the calling party. Upon receiving the URL in response from the Subscriber and Service Data Repository, the OnNoAnswer is being called in, posting the URL retrieved from the OnNoAnswer plugin to the callers client.

10.2 Use Cases

The OnNoAnswer service is a terminating service, being activated when the called party do not answer an initiated call after alerting, or when the subscriber has instructed the gatekeeper, based on some known preconditions, that the call should not be routed to his end-point, but being terminated at the gatekeeper. There are two central actors, the caller and the called party. The caller can be the subscriber of the service or not. The actor caller is initiating a call, proceeding a call, or if he is the subscriber of the OnNoAnswer service; maintaining the service. The use cases are described in figure 10.2.

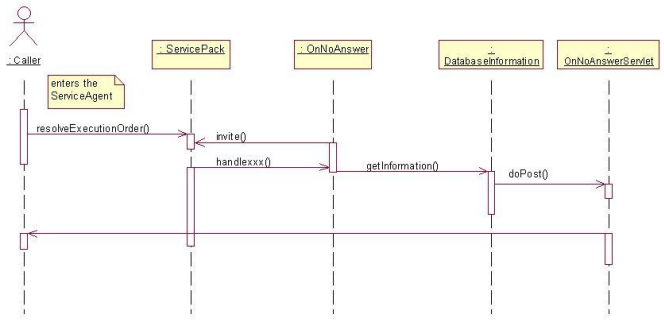


Figure 10.3: Sequence digram for the initiating of a call to the OnNoAnswer service

10.3 Resolving the execution order

A subscriber can have several different services active. The resolving of the execution order is done in the Service Pack. The Service Pack implemented in this solution was a minor Service Pack implemented just to run the one service. This Service Pack was configured to run in the Service Agent making sure the service was called in on the right conditions. There were no problems during this phase of implementation.

```
The invite method is the central method in the Service Plugin. The invite method te
```

```
public boolean invite(int eventId, int callDirection){
 if(eventHandler.DEBUG)
    eventHandler.debug(SaDebugGroup.SERVICEDATA, 9, "OnNoAnswer",
                           "invite", "invite of OnNoAnswer plugin");
 switch(eventId){
  case ServiceInteractionController.EVENTID_INVOKE:
  case ServiceInteractionController.EVENTID_NO_ANSWER_INTERCEPT:
  case
  ServiceInteractionController.EVENTID_CALL_RELEASE_CALLEE_INTERCEPT:
    if (callDirection==TERM_CALL)
    return true;
    else
    return false;
 default:
    return false;
  }
}
```

The ServicePlugin returns true to the Service Pack if the conditions in the call setup is right. Further the ServicePlugin then acts on this conditions. In the handle invoke the service Plugin ensure that we are notified on a no answer event og if the called party is busy.

```
public ServiceActionRequest handleInvoke(){
   CallInfo callInfo = getCallInfo();
   ServiceActionRequest sar;
   .
   .
   .
 return sar;
   }
```

The following procedure resolves the execution order, all other parameters being defined.

```
//kboPackKode

private static final int [][]EO_INVOKE = {{},{ OnNoAnswer.SPID}};
private static final int [][]EO_NO_ANSWER_INTERCEPT = {{},
  {OnNoAnswer.SPID}};
private static final int [][] EO_CALL_RELEASE_CALLEE_INTERCEPT = {{},
  {OnNoAnswer.SPID}};

//Resolve the executiuon order for INVOKE and NO_ANSWER_INTERCEPT.
//All others events are of no interest for his prototype

protected int[] resolveExecutionOrder(int eventId, int callDirection)
{
 switch (eventId)
 {
 case ServiceInteractionController.EVENTID_INVOKE:
    return EO_INVOKE[callDirection];

 case ServiceInteractionController.EVENTID_NO_ANSWER_INTERCEPT:
    return EO_NO_ANSWER_INTERCEPT[callDirection];

 case ServiceInteractionController.EVENTID_CALL_RELEASE_CALLEE_INTERCEPT
    return EO_CALL_RELEASE_CALLEE_INTERCEPT[callDirection];

 default:
    return null;
 }
}
```

As shown in the code-example above the OnNoAnswer service is isolated and is called in on invoke and if there is no answer. The result is returned, and the OnNoAnswer plugin is called in, by the fact that the OnNoAnswer Service have told the Service Pack that is wants to be called in on invoke and no answer event if it is a terminating half call.

```
//The invite method tells the Service Pack if we
```

```
//want to be called in or not.
//In this case we want to be called in on the
//INVOKE and on NO_ANSWER_INTERCEPT
//events if in a terminating half call, else not.
//i.e releaseComplete, busy, no answer*/

public boolean invite(int eventId, int callDirection)
{
 if(eventHandler.DEBUG)
    eventHandler.debug(SaDebugGroup.SERVICEDATA, 9, "OnNoAnswer",
                           "invite", "invite of OnNoAnswer plugin");
 switch(eventId)
 {
  case ServiceInteractionController.EVENTID_INVOKE:
  case ServiceInteractionController.EVENTID_NO_ANSWER_INTERCEPT:
  case ServiceInteractionController.EVENTID_CALL_RELEASE_CALLEE_INTERCEPT:
    if (callDirection==TERM_CALL)
    return true;
    else
    return false;
 default:
    return false;
  }
}
```

Further more we have to tell the Service Pack that we wants to be notify upon a no answer event. Since telephony is time sensible I have choice to go towards the database as son as the service is called in. The negative consequences of this solution is that is could result in wasted request towards the Service & Subscriber Data Repository (SSDR), but this design-decision would require some actual testing before it could be measured both in cost and load of the system.

```
//In the handle invoke we will ensure that we are notified on a no
//answer event  and if B is busy.
//We return a ContinueCallRoutingSar. This is
//constructed without servicePluginID,
//a CallRoutingInfo and the monitor set as parameters.

public ServiceActionRequest handleInvoke()
{
  //System.out.println("WIO -- User: " +
  getUser().getAliases()[0] + " is
  connecting to " + callInfo.getDestinationAliasInfos()[0]);
  sar = new ContinueCallRoutingSar(SPID,
        new CallRoutingInfo(getCallInfo()), mons);

  return sar;
```

```
}
```

During design and prestudy it was decided to use a HTTP Servlet to use information from the ServicePlugin to generate a html-page to return to the calling party. When following this approach the ServicePlugin would do a look-up i the database, and pass the information along to the Service Servlet, which would tag the information and post it to the calling party.

10.4 Retrieving Information

The Ericsson IP Telephony system holds a database for all service and subscribing information related to a call session. The database, and the design of this has changes several times, facing larger grand changes during the implementation of the OnNoAnswer Service. The database solution I have chosen to use is the Subscriber and Service Data Repository. This is a database based on technology solutions from Oracle and Jigsaw. Since the OnNoAnswer Service is implemented as a protoype only, the actual adjustment to the database where not done.

When designing the OnNoAnswer some effort was make to analyze how a possible organization of the data could be done, the NIAM structure of this can be viewed in Appendix B. The requests to the database would work as for other services in the Ericsson IP Telephony system.

The result of the request to the database is a xml-document. This is sent to the OnNoAnswer Servlet who by means of parsing algorithms identifies the informations, converts this to html, and post it back to the calling party.

When the call is first initiated the OnNoAnswer finds the right service profile based on the e-mail identifier identifying the user in the IP Telephony system. This service profile is loaded back from the SSDR and cached in the Service Agent. Further more the identifier identifying the calling party is sent to the SSDR, for identifying the calling party to retrieve the correct URL to be loaded back to the calling party. Based on the information in this return service profile, the OnNoAnswer plugin, is sending the URL to the OnNoAnswer Servlet that posts this to the calling party.

If the calling party is identified and authorized as the subscriber the URL returned are the configuration URL that maintains the subscribers configuration of the service. This is posted back to subscriber. When the subscriber have finished editing his configuration the new information is sent to the OnNoAnswer Servlet. The Servlet reads the parameters updated, and sends a notification to the SSDR.

```
public static void getInfomationFromDatabase(String called, String caller
```

```
 {
  String browserName = "c:/test_explorer/IEXPLORE.EXE";
  URL  url;

  // if B_no == A_no the subscriber wants to
  //update his subscription page,
  //The look up in the database will return
  //B_no.subscriptionPage
  // with all lists editable for the subscriber.

  if (called!=caller)
  {
    // subscriber
    url = subscriberProfile.getProfile();
  }
  else
  {
Alternatively    //caller
    url = subscriberProfile.getprofile(caller);
  }
  serviceServlet = ServiceServlet.getOnNoAnswerServlet();
  if (serviceServlet==null)
  {
    // something wrong - give notification
  }
  else
  {
    serviceServlet.handleUrl(url);
  }
  if(eventHandler.DEBUG)
      eventHandler.debug(SaDebugGroup.SERVICEDATA, 9, "OnNoAnswer",
                         "handleInvoke",
                         "invoke of web-page +in OnNoAnswer plugin");
    try
    {
      Runtime.getRuntime().exec(new String[] {browserName, url});
    }
    catch (IOException exc){}
    exc.printStackTrace();
    }
 }
```

10.5 Posting of the service

The next part of the service is considering the information and options presented to the calling party. Since the prototype is based on a scenario where the call is initiated from a PC-phone, for instance by the means of some click-to-call functionality embedded in a web-browser, the posted call

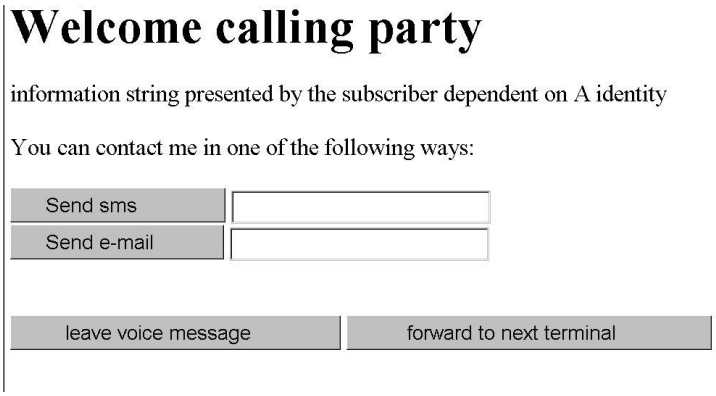


Figure 10.4: Web page presented the calling party

back is designed for a web-browser on a PC. This interactive web-design is a prototype design that is not covering all the aspects of interactive web-design, but is designed to mainly cover the functionality the service offers in a easy and intuitive way.

Since the service should, and likely would, be used by both calling parties and the subscriber, two different angles are of interest; interaction for updating the service information and option by the subscriber as opposed to the operator, and interaction for further handling the call for the calling party.

The calling party is presented a web-page, giving him options and possibly some information explaining the whereabout of the called party, and how to further process the call.

In this case the calling party will be presented with several option on how to proceed the call, depending on what option that has been authorized to the group of callers he is identified as.

For illustration, let say the calling party is located in the group of Friends & Family, and he is presented the options as shown in figure 10.4:

- Send e-mail. If the calling party chooses his option an email will be sent to the address specified by the subscriber. This address could be hidden from the caller. the e-mail address can be sent to the caller via a mailto-tag, and a separate mail application will be launched. Then a call release will effectuate.
- Send SMS. If the calling party chooses this option, a SMS message will be sent from the web-browser to the cellular terminal, often a mobile phone, defined by the subscriber.The SMS address can be hidden from the caller.
- Leave voice-mail. If the calling party chooses this option he will be

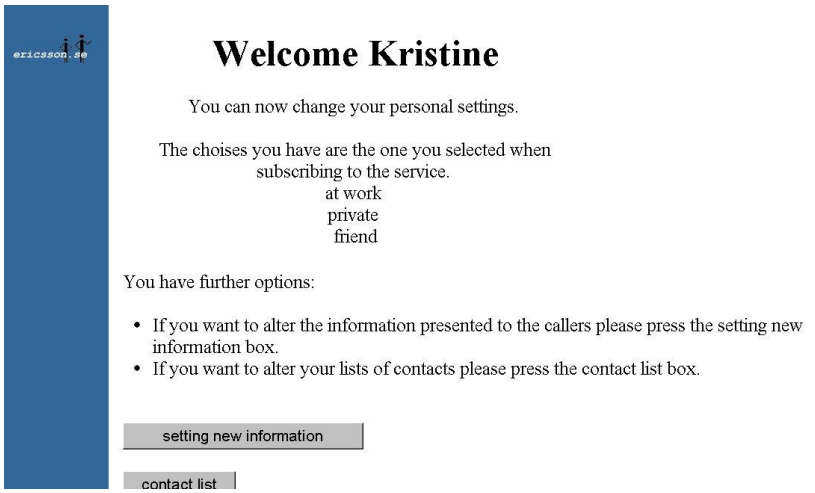


Figure 10.5: Illustration of the HTML-page presented to the subscriber

forwarded to the subscribers voice-mail. The voicemail address will be sent to the caller, but can still be hidden.

- Forward to next terminal. If the calling party chooses this option, he will be forwarded to the nest terminal registered in the system by the subscriber, or an operator.

If the calling party is the subscriber, he will be presented with options for altering the information string being presented to the callers, or altering the contact lists. This modification will, when the subscriber presses the "Ok" button, be posted back to the OnNoAnswer Servlet, which reads the updates and sends this back to the OnNoAnswer plugin as database objects, which in turn is written to the database. Figure 10.5 gives an idea of how this grafical user interface could look like. If the calling party is identified as the subscriber of the OnNoAnswer service, the service will reply with a web-page containing the subscribers configurations, giving him the options for altering the content being presented, and adding and removing contact from his lists.

For testing the service an HTML-page were sent directly to a known IP address and port with an execute statement and by posting a minor servelt, not giving the user actual possibility for proceeding a call.

Depending of the choice made by the calling party different call handling is required.

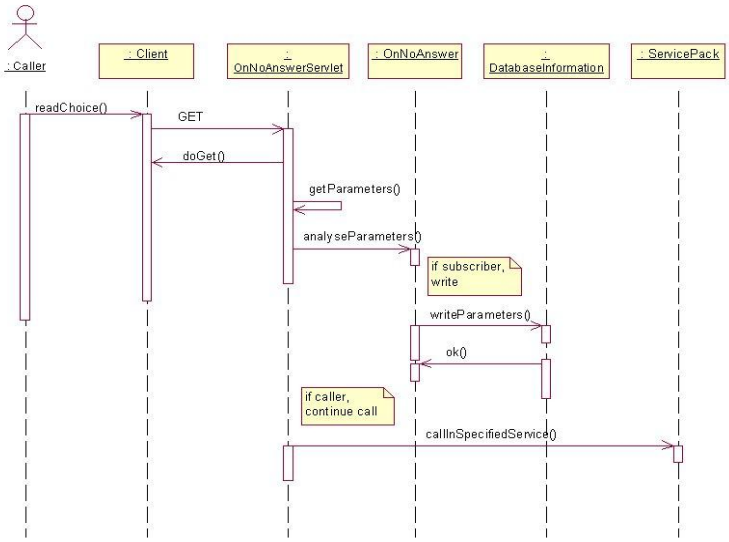


Figure 10.6: Sequence digram for the proceeding of the call

10.6 On proceeding the call

Figure 10.6 shows the interactions between the different modules in the OnNoAnswer service when the user initiating the call is making a choice on how to proceed the call. The identification of the initiator of the call done here, is not actually necessary, since this information is given previous, and could be held by the service waiting for the response.

When a call is initiated and identified, the Service Pack has to define in which order the different service should be called upon. This could be stated inside the actual network, or it could be defined by the operator. For the specific Ericsson IP telephony solution this is defined in the system. By means of this the Service Pack defines static the execution order. To illustrate this with a small example lets review the Personal Agent Service again, and identify possible execution mismatches.

If a subscriber have activated both the personal Agent and the On-NoAnswer Service the system must identify and prioritize which service should be called in first. Recall that the Personal Agent forwards the call to the other terminal registered. That is, if the subscriber has registered himself in the system with both a cellular phone and a PC phone, where the latter has the lesser priority, the call would first be forwarded to the cellular phone, for later to be forwarded to the PC phone upon no reply situation.

If the OnNoAnswer service is activated it will then be called in if both option in the personal Agent failed. This is the same, as saying that

the subscriber will take all his calls on one of the two, and only in case of situation where he will or cannot answer the OnNoAnswer Service will be called in. This is a simple operational solution, but it is not taking into consideration that the called party would like to reserve himself from some calling parties. To be able to do this, with the execution order described above the called parties identification lists over friends & family, work and other, and the status of forwarding options must be declared before any of the services is called in. This would then mean that the listings could not be located in one separate service. Before the execution order could be defined, there should be some logic deciding upon which service who had priority over which listings. This combining of services is very common, and adds grater flexibility and possibility for more tailored solution for each subscriber.

Nevertheless; Due to restricted access to the Ericsson IP Telephony system and its design when implementing the prototype I have chosen to isolate the OnNoAnswer Service totally. This means that some important aspects lacks, but it gives the opportunity to evaluate the service isolated before implementing further.

The database solution for this prototype was for the database to hold both the information strings and the contact list and the options for further processing of the call. These variables could then be selected and coordinated by the logic identifying the calling party. By doing it this way, the OnNoAnswer plugin would get an URL from the database, and the OnNoAnswer servlet would post this URL to the calling party. This leaves logic and execution to the database, possibly slowing it down. Another solution were for the database to send the variables as separate objects. Then the OnNoAnswer servlet would generate the HTML-page and post this.

10.7 Problems during the implementation phase

The implementation took place during the autumn 2000 at Ericsson. When I started the implementation I had not enough knowledge of the system, and time were spent on getting to know the Ericsson IP Telephony system. During this phase both the database structure and the ClearCase source control structprototypingure changed, and parts of my project got corrupted.

Putting the pieces of the project together took time, and the work with the prototype were re-defined, only to include the implementation of the plugin and servlet. The work conducted on the database aspect were minimalised, and the initial plans on conducting a test of the OnNoAnswer service both in terms of alternative ways of implementing it, and getting input from users were discharged.

Due to these problems I decide to end the implementation phase and document the work done.

10.8 Limitations in the implementation of the OnNoAnswer Service

Today, the Ericsson IP Telephony system using H.323 do not support sending port numbers and IP addresses to identify the web browser/web server in the call set-up messages. By means of this, there is actually now way of knowing where to send the posted response. If a field in the call set-up messages where to be used for sending port numbers and IP addresses, it is standardized in H.323 version 4. To come around this time-consuming process it is possible to use a standardized, non-used field for text only. Then some other logic must be implemented to identify, separate and validate legal IP addresses and network identifiers, as well as port-numbers. This information must be passed all the way from the Sitekeeper and the User Agent to the Service Agent, since the posted reply is initiated directly from the Service Agent. Due to the nature of this , I did not chose this solution, since it would have taken a lot longer time to implement.

The prototype is illudating a known IP address and port number, and do not further problematize this.

To finish the OnNoAnswer Solution handling of the service must be implemented in the SSDR, or an entire new test-database must be developed. Some work has been done, but due to lack of time the work was not finished. The database is based on Oracle and xsql and xml. The idea is to send a request to the database. This request will result in a response generating an xml page. By the means of parsing-algorithms the elects in this page can be parsed and sent to the OnNoAnswer Servlet. After performance testing, it should be analyzed if it is more sensible to return a full URL to launch directly. Then the database must be able to generate and hold full html-pages, or pages based on cascading style sheets. This database work is central, but time-consuming, and it was not time to do this work.

Further more the interaction with other services provided must be clarified. As the prototype was developed I only looked into this one service as a subscribing service. When the user should then choose for further processing of the call, interaction with other services came in question. As it is today, some services are hierarchal placed to dominate others. This is not always by the choice of the subscriber. Even when I for the means of prototyping set the OnNoAnswer service as the first service to be called in, interaction with other services must be analyzed and designed for the further processing of the call.

Further more it should be discussed what kind of logic that could be

left to the terminal, and what logic that should be implemented in the network. Before doing this, one must look closely at future terminals.

Taking into accounting the lesser success of the UPT service, this prototype do not consider identification of the service-owner. For the prototype implementation of the service the subscriber calls his own number, and are identified by the Ericsson IP Telephony system. By matching the a-number and b-number, he will not be connected to his own services on busy events, but called in on OnNoAnswer service, requesting the SSDR to retrieve his editable web-page that are posted back to him. He can here do the editions he wants, and post them back. The changes and updates will be written to the SSDR and effectuated.

Chapter 11

Usability Testing

As new services is developed and employed rapidly, the need for “the experienced user” emerges. By this I mean that the applications and services offered to the end-user demands that the user have an understanding and the expertise about how to interact with communication and data applications.

The famous story about the middle-aged lady who wrote the answer to competition about a trip to South-Africa on a note, put it in an envelope, and wrote the e-mail address to the Norwegian Television Company (NRK) on the envelope and put in in the mailbox is well-known, and often giggled at. But is it really amusing, if not sad? The e-mail was presented as an address, and the user used this address in the way she was used to, by means of the national posting service. This is an example on interacting with new application, and that these are not always self-explained. More important it shows the importance of thinking in terms of usability when designing new applications.

During design and implementation of a service like the OnNoAnswer service described in the previous chapter, interaction between the end-user and the application is essential. The interaction designed in the prototype takes place via a Graphical User Interface presented by means of a browser on the end-users terminal. I will therefore describe some of the aspects of interactive web-design and usability.

In implementing the prototype the functionality was in focus. The prototype was developed both to analyze the functionality as described, but also to see if this supplementary service embedded functionality that would make the service a value-adding service with regards to being useful for actual use in call handling, offering a new feature to the user, making him in charge of his own availability. The prototype was also developed to get a feeling for what technologically obstacles that where to be solved, and discover other solutions and implementation possibilities.
might
Furthermore it is good design principle to develop a prototype to get some

input from the users about the needs and usage of a future service. By asking and consulting the users, the designers get valuable input and corrections, and it is more likely that the service developed will be a success with lesser maintenance and support work.

Users expectations and needs of a service will change over time, and it is reasonable to believe that a service designed to make the users in charge of their own availability will change as the society changes. Never the less will the service also meet new demands when the technology changes, and new possibilities opens. This interaction between society and technology will mark the development and deployment of new services. When designing a new service, it is important to have in mind both the needs of todays society and future trends, and the possibility that lies in future technology.

11.1 Designing the interface

I will describe some of the aspect when designing an interface for user interaction with applications as the OnNoAnswer service. According to the prototyping method I have adopted, I have used a rapid format for presenting the information to the user, plain HTML, see figures in chapter ?. Never the less, the same principles would adjust when the prototype is modifies and is implemented as an actual application. Therefore the design principles illustrated below adopt to both the Graphical User Interface (GUI) in the prototype and the application evolving from this prototype.

The OnNoAnswer service should in all cases be displayed in a web browser to the end-user, and interaction with the service should be operated from here. On important issue arises when designing an applet. Should the applet be displayed in the end-user's current browser, or in a browser window? The current browser is well-suited for displaying applets in which users performed a single task. This approach enables users to perform the task and then resume other activities in the browser, such as surfing the web [JLF]. A disadvantage of using the current browser window is that the applet terminates when end-users navigate to another web-page. The current settings and data in the applet are lost. To use the applet again, users of the OnNoAnswer applet/prototype, must make a new call setup.

If the applet designed involves more than one task or if users might visit other web-pages before completing the task, [JLF] suggests that a separate window should be launched for displaying the applet. This approach enables users to interact with the applet and maintain the original browser window for other activities.

Since the on OnNoAnswer service offers several option for proceeding a call, the end-user might want to for example both send SMS and leave voice-mail. This could be implemented in the service logic so that all

options is active until the end-user manually closes the window, and that the closing of the browser window is the key for ending the session. These multiple aspects should be tested with end-user, but for the initial prototype I suggested that the response on a call initiated should be presented in a separate web-browser.

In Software design, accessibility requires taking into account the needs of people with functional differences, for example users who are enable to operate a conventional mouse or keyboard or users who cannot process information using traditional output methods. Means for designing an application to and support this is for example by providing accessible names and description, so that a screen reader could interact with the component.

A blind person, for example, want a voice-based menu, instead of an HTTP-browser. By using XML and style sheets the conversion from web-page to voice-instructions might be automated. For deaf persons however, web and SMS and other textual mechanisms have opened up new ways of communication much easier for them to use than the old "text-telephony-service-for-the-deaf".

One should also plan for internalization when designing an interactive application. By using layout managers and recourse bundles, all components and text will adjust to different languages, and be resized.

The resizing of components are as important as the choices made to what component to use for representing different types of information, and interaction components. In application on the Internet today, there are many different ways of inert-acting with for example multiple choices from a list or a table. It is important to use the component in the application, that is the buttons, list and tables, with the component functionality best suited the needs of the end-users and the functionality of the service being offered. [JLF] presents a diversity of the components and use of this when designing a Java application like the OnNoAnswer prototype.

Chapter 12

Conclusion

The motivation for the work described in this paper is the introduction and discussion of Annex K, a supplement to the H.323 standard for IP Telephony. This draft describes a way for employing new services in an IP Telephony system by means of the Hypertext Transfer Protocol.

By looking into existing networks and future technology, the concept of a new service based in an IP Telephony network system has been analyzed and prototyped. The service in question is a supplementary service related to call handling. That is, the service is designed to give the users more flexibility and control over traditional call processing.

I have in this paper described some existing networks, PSTN and GSM, and some future networks, GPRS and UMTS. Taking the architecture of these networks into consideration I have presented some existing services and supplementary services based on the possibilities these networks and architecture design offers. With this as a basis I have looked closely into IP Telephony structure and service. I have presented IP Telephony systems, based on the H.323 standard in general, and further more presented an actual IP Telephony system implementing this, the Ericsson IP Telephony system.

Based on the design of this system I have analyzed and prototyped a supplementary service based on the upcoming H.323 standardization Annex K, which provide an optional way for controlling user directed supplementary services in an H.323 environment. The core of the service implemented, the OnNoAnswer service, is an alternative way of responding to events when the called party is not answering, or do not want to answer an initiated call.

The OnNoAnswer service offers the user means for an optional way of proceeding the call, that is by sending e-mails, SMS, connection to voice-mail or other devices. For the called party, the subscriber, this service offer grater flexibility, control and screening capabilities.

12.1 The functionality

The main functionality of the OnNoAnswer service is to let the subscriber of the service be in charge of his own privacy and time. The OnNoAnswer service is in this prototyping mainly been prototyped for personal use. The service has been analyzed and designed for use outside the working environment, being focused on use for personalized management.

The actual functionality resembles both to the Personal Agent service in the Ericsson IP Telephony system, on the iPulse service and the UPT service being described in previous chapters. The follow-me aspect of the OnNoAnswer service resembles different forwarding conditions already existing and supported in most networks.

These options link different traditionally networks together, combining traditional tele services and data services, as in connection to voice-mail or e-mail, is a new feature. Further more is the aspect of combining routing to the end-point or letting the gatekeeper be in charge a new possible way of combining network centric and terminal centric views of looking at service provisioning.

The major drawback of the service is that it requires a rather cumbersome updating procedure for the subscriber of the service if it should e of any use. Operating with lists of contact, requires that the subscriber updates and edit these list.

To make these updates less cumbersome, multiple ways of updating the list should be offered to the subscriber, it should be possible to transfer the whole phone-book from the subscribers cellular phone to one list, all called number on the subscribers work-phone in another list and so forth. This should be done in one operation, controllable from all known devices, for example fixed phone, mobile phone or PC.

Authentication of the calling party is done in the Access Broker when the calling party is entering the system.

12.2 Technological aspects

The implementation of the OnNoAnswer prototype revealed several technical aspects to be discussed and tested.

With the adding the functionality described in H.323/Annex K [ITUTD-13] identifying the address on which to post the response will be solved. The address will be sent via the call control API from the User Agent to the Service Agent, that gives this information to the OnNoAnswer service, when on responding on the call will post the right URL based on the pre-conditions.

Another technical issue to discuss, is when the OnNoAnswer is invoked, it will always send a release call due to the refusing of answering the initiated call. When the initiator of the call, the calling party, wishes to proceed the call by being diverted to for example voice-mail, a new call leg is set up, and an entire new call session is being initiated. Since the cost of setting up a call, in most systems, is higher then continue a call, this is not a good solution. On the other hand, if the calling party wishes to terminate the call upon reception of an URL, he must be informed that the call is proceeding until he ends the session, that is; closes the window. Then a release call will be initiated and the call is teared down.

One solution is to introduce a new ContinueCallRoutingSar. This response should inform the User Agent that the response is a OnNoAnswer-URL, and although this URL is posted by other means then the H.225.0 message, the H.225.0 message should hold information, so that billing could be stopped until a new event is received from the calling party, and only then should the billing start up again. This could imply a lot, non-active and non-profitable traffic in the system. But it gives the calling party the means of processing the one call he actually initiated.

Another aspect is who should pay for the traffic: the calling party or the subscriber of the OnNoAnswer service? Since the calling party may not be aware that the called party do subscribe to the OnNoAnswer service, he may not like to pay for further processing of the call, here the posting of an URL, and he will not pay for this. If the called party is not reachable, then the call session should end. However: posting the URL and getting the menu may well be free of charge for the calling party, since this is the web and may be by commercials, much like hotmail is today. If calling party does not want any Annex K functionality, he may ignore the URL and end the call session.

Another aspect of the OnNoAnswer service, which also employs for other supplementary services, is how the service should be deployed in the system. The prototype is implemented directly as a plugin in the Service Node in an IP telephony system. By means of different environment offered by the Service Agent server the OnNoAnswer prototype service is plugged into the server. As the service is classified as a supplementary service it should be possible for a service provider to design, implement and offer the end-user services like the OnNoAnswer service, without depending on the service existing in the network the service provider is buying network recourses from.

If independent service providers should be able to add services based on call control information, this call control information must be offered through a well-defined interface or API to the service provider.

12.3 Further work

Users of the service are both the subscriber to the service, and the users who tries to contact the subscribers. These two different categories of users will have different needs for usability. The subscribers are the one who are managing and maintaining the information presented in the service. One hypotheses based on the experiences with the UPT-service and Alfa-number is that the service should not be complex to maintain and manage. That is; I believe before testing the OnNoAnswer service that the subscriber to this service will want an interface to maintaining the service that is fast, easy and with little room for making errors.

Based on this the usability testing will be divided into two:

- the subscribers
- the callers

When testing the OnNoAnswer service on these two categories there are several topic of interest. Most important is the question on whether this is a service worth subscribing to and paying for. What good will a service with the features of the OnNoAnswer service bring to a subscriber in managing his own availability?

Second; how is the service filling the needs of the callers? When the subscriber is not answering the call, what further option for proceeding the call is relevant? Getting these choice on a web-page for sending an e-mail directly, instead of opening the callers own e-mail application and phone book, is it more likely that the caller will send an e-mail, leave a voice-mail or ask for transferring the call to a colleague.

The question stated above are more fundamental than how a button is placed or how the text is colored. Usability of a service is both about getting the right information and options at the right time as it is getting them presented in a readable and understandable way. Bad Grafical User Interface (GUI) design could corrupt the underlying questions, and make a service appear more complex and difficult to use than was intended. To summarize; how to manage information, setting values and making choice are as important as readability of the information, navigating between different choices and getting understandable responses to actions performed.

Since the database gives the information necessary for presenting the calling party with the information, and the option he needs in a xml-format, the service can easily be employed to several different devices. Style sheets can be employed to adjust the respond to any device supporting use of tags.

It could be further developed to support end-points supporting WAP, in addition to HTML. For the servlet to know which format to post, the

endpoint may negotiate media, and identify its own resources to the service.

This negotiation of media is central in the IP telephony system. For the development of the prototype above I have assumed that the call goes from a PC-phone to any other end-point. But all though the assumption of the calling party initiating the call from a PC-phone is central for narrowing the areas that must be discussed and covered in this thesis, it is not a realistic one in a real system. It does not make much sense to post a video to a GSM cellular phone, event thou the network in between can handle the transmission. The Open Service Architecture (OSA) contains terminal capabilities that may be of use to such services.

Another option is that the Terminal Agent (TA) in the Ericsson IP Telephony system adopt to different media. In this way the core service logic will not need to know if the user is on WAP-device or an full-blown PC. Translations from web-pages to reading menus vi a a (programmable) Interactive Voice Response (IVR) can also be done near the TA, that is the access site. A WAP gateway would also be located at the same access site. This means, that the media negotiation could take place both at accessing the network, that is the access site negotiates the media, not involving the services.

Bibliography

[Aud99] Jan A. Audestad. *Distributed Processing in Telecommunications*, Course Material in DIPT 1999, at UNIK Kjeller, 1999.

[BASHT] Selena Sol. *The Basics of HTTP*, *http://www.wdvl.com/Authoring/HTML/Tutorial/http.html*, April 2001.

[BUCK] Simon Buckingham. *An Introduction to GPRS*, *http://www.mobileipworld.com/wp/wp3.htm*, April 2001.

[BUD] Budde R., Kautz K., Kuhlenkamp K., Zullighoven H. *Prototyping: An Approach to Evolutionary System Development*, 1991.

[CIS] Cisco Systems Inc. *Quality of Service Networking*, *http://www.cisco.com/univercd/cc/td/doc/cisintwk/itodoc/qos.htm*, April 2001.

[CSJSP] Marty Hall. *Core Servlets and Java Server Pages*, Sun Microsystems Press, Prentice Hall PTR 2000.

[Dat98] Jiri Kuthan, FOCUS-mobis. *A Primer on the H.323 Series Standard*, Mai 1998. *http://www.fokus.gmd.de/research/cc/glone/projects/ipt/references/frameset.php3*, April 2001.

[DIP] Lill Kristiansen. *OSA in 2G and 3G*, Course Material in DIPT 2001, at UNIK Kjeller 2001.

[GLASS] Steven Glass. *Telecommunication Systems: An Introductory Guide*, *Gilbert Tobin*, 2000 *http://www.gtlaw.com.au/pubs/telcosysintroguide.html*, oktober 2000.

[GSMWORLD] GSM Data Services. *http://www.gsmworld.com/technology*, April 2001

[HER] Do van Than, Børge Nilsen and Jo Herstad. *Unifying real time communication services over Intranet*, Paper submission to the Globecom'99.

[IC99] Internet.com *The Basic of HTTP*, *Internet.com Corp.Legal.Notice.Privacy Policy*, April 2001

[IECA] International Engineering Consortium. *Application Services Framework*, *http://www.iec.org/tutorials*, April 2001.

[IECIP] International Engineering Consortium. *Internet Telephony*, *http://www.iec.org/tutorials*, April 2001.

[IECS] International Engineering Consortium. *Signaling System 7*, *http://www.iec.org/tutorials*, April 2001.

[IECW] International Engineering Consortium. *Wireless Internet Network Communication Architecture http://www.iec.org/tutorials*, April 2001.

[IER] Official iPulse site. *http://www.ericsson.com/ipulse/*, April 2001.

[IETF97] Bernes-Lee, T. Fielding R.T. and Frystyk Nielsen H. *RFC 2068: Hypertext Transfer Protocol - HTTP/1.1*, *IETF Internet Engineering Task Force*, January 1997.

[ITU98a] ITU. International Telecommunication Union. *ITU Telecom Information Exchange Services http://www.itu.int/home/Aboutsite/index.html*, 2000.

[ITUTD-13] Update of H.323 Annex K. *Document TD-13, Portland*, ITU Telecommunication Standardization Sector Study Group 16, August 2000.

[ITU2000] ITU-T. *H.323 – Packet-based multimedia communications systems*, Series H: Audiovisual and Multimedia Systems. International Telecomminication Union, draft version 4, October 2000.

[ITUMO] Sengodan S., Koodli R., Rajahalme J., *Wireless and Mobility Issues in IP Telephony* Nokia Research Center, Burlington USA, October 1999

[IPTER] *IPT 2.0 Presentation*. Ericsson AS, summer 2000, For internal Ericsson use only!

[IWP] iPulse White Paper. Promotion and sales toolkit, Internal Distribution, september 1999.

[JLF] Sun Microsystems, Inc. *Java Look and Feel Design Guidelines*, Addison Wesley Longman, Inc, June 1999.

[KHN99] Lill Kristiansen, Jan Holm og Borge Nilsen. *Splitting the GK and some reasons to do so.*, *Tiphon 13*, 30. April 1999. L.M. Ericsson.

[KLI] Kevin Kliland. *Teknologiske Utfordringer for VoIP/IPT*, Ericsson Presentation, Oktober 2000.

[KOR] Markku Korpi and Vineet Kumar. *Supplementary Services in the H.323 IP Telephony Network*, *IEEE Communicationtions Magazine*, July 1999

[MAT] Mathiasen, L. Munk-Madsen A., Nielsen A. P., Stage J. *Objekt-orientert Analyse og Design*, Forlaget Marko ApS, Aalborg 1998.

[NIE+2000] Jacob Nielsen. *Designing Web Usability*, New Riders Publishing, indiapolis, Indiana USA, 1999.

[NILS+1999] Arild Nilsen. *Service Creation Environment - Developers Guide*, Ericsson ETO, Open description, December 1999.

[NILS+2000] Arild Nilsen and Kevin Kliland. *Subscriber and Service Data Repository* Ericsson Limited Internal, February 2000.

[PD96] Larry L. Peterson and Bruce S. Davie. *Computer Networks: A System Approach* Morgan Kaufmann Publishers Inc, 1996.

[PRO] Lill Kristiansen. *prOsa: proof of concept of OSA*, Document Number: 0363-FCPR 103 030 UEN, Telenor and Ericsson AS, December 2000.

[Ros99] Rosenberg J., Lennox J., Schulzrinne, H. *Programming Internet Telephony Services*, Tech-Report Number CUCS-010-99, October 1999.

[SCH99] Henning Schulzrinne. *Converging on Internet Telephony, Service for Telecom Version 2.*, *IEEE Internet Computing*, page 40–43, Mai-Juni 1999.

[SCSS] WWW COnsortium. *Specification of the Cascading Style Sheet Standard*, *http://www.w3.org.pub/WWW/TR/REC-css1*, May 2000

[SKO] Endre Skolt and Ivar Oldervik. *Intelligent Networks as a Platform for Provisioning of Services in GSM and DECT*, Teletronikk Volume 91 No. 4, Oslo 1995.

[SOM] Ian Sommerville. *Software Engineering*, Addison Wesley Longman Ltd, ch 8, 1998.

[SUN] Java Develeoper Connetcion. *Fundamentals of Java Servlets*, *http://developer.java.sun.com/developer/onlineTraining/Servlets/Fundamentals/servlets.html*, April 2001.

[TAHN] Do van Thanh. *Compendium in MOBKOM: Mobile communications*, UNIK, 2000.

[TIR] Stefan Tirtey and Bijan Jabbari. *An Analysis of Advanced ISDN Supplementary Services*, *http://mozart.gmu.edu/tmp/paper.html*, August 1995.

[TOG] James Toga and Hani ElGebaly. *Demystifying Multimedia Conferencing Over the Internet Using H.323 Set of Standards*, *http://www.intel.com.tw/technology/itj/q21998/articles/art_4a.htm*, Intel technology Journal, 1998.

[UMTSFORUM] UMTS Forum. *Enabling UMTS/Third generation Services and Applications*, No. 11 Report from the UMTS Forum, October 2000.

[WAL] Knut Erik Walter and Per Hjalmar Lehne. *The Basic of Mobile Comminications*, Teletronikk Volume 91 No. 4, Oslo 1995.

[3GPP] 3GPP specification. *Working Draft 3G TS 23.228 v. 1.7.0*, *http://www.3gpp.org/ftp/Specs/Latest_drafts/23228-170.zip*, April 2001.

Appendix

Appendix A

HTML-code

A.1 HTML-page presented to the subscriber

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
                  "http://www.w3.org/TR/REC-html40/loose.dtd">

<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=windows-1252">
<TITLE>Service update</TITLE>
</HEAD>


<BODY LINK="#000000" VLINK="#000000" BACKGROUND="Image4.gif"
ALINK="#000000">

<BODY BGCOLOR="#FFFFFF"
background="http://people.ericsson.se/gifs/blue-white.gif"
LINK="#000000" VLINK="#000000" ALINK="#000000" >
<TABLE border=0>
    <TD width=150 valign=top align=left>
<img People logo" src="http://people.ericsson.se/gifs/logo.gif">
    </TD>

<TABLE CELLSPACING=0 BORDER=0 WIDTH=623>

<! dette bør være dynamisk>
<h1>Welcome Kristine</h1>

<p>
You can now change your personal settings.
</p>
<p>
```

```
The choses you have are the one you selected when s
ubscribing to the service.
</p>
<!her kommer tabellen brukeren kan ender på>

<br><input type=checkbox class=nxs name=alt_svar value=3766> at work
<br><input type=checkbox class=nxs name=alt_svar value=3768> private
<br><input type=checkbox class=nxs name=alt_svar value=3767> friend
</form>
</table>

<p>
You have further options:
<p>
<ul>

<li>If you want to alter the information presented to the
callers please press the setting new information box.<br>
<li>If you want to alter your lists of contacts please
press the contact list box.<br>
</ul>
<br>

<FORM METHOD="post" ACTION="http://www.stud.ifi.uio.no/~kribolst">
<!når han trykker her skal ha sette ny personlig informasjon>
<input type=submit value="setting new information"> 
</form>

<FORM METHOD="post"
ACTION="C:\Jigsaw\Jigsaw\WWW\User\Abonnent\contacs.html">
<!når han trykke her skal han få opp kontaklisten>
<input type=submit  value="contact list"> 
</form>
<br>

</BODY>
</HTML>
```

A.2 HTML-page presented to the caller if not the subscriber

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
                  "http://www.w3.org/TR/REC-html40/loose.dtd">

<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=windows-1252">
<TITLE>Service update</TITLE>
</HEAD>

<BODY LINK="#000000" VLINK="#000000" BACKGROUND="Image4.gif"
ALINK="#000000">

<BODY BGCOLOR="#FFFFFF"
 background="http://people.ericsson.se/gifs/blue-white.gif"
LINK="#000000" VLINK="#000000" ALINK="#000000" >
<TABLE border=0>
    <TD width=150 valign=top align=left>
<img People logo" src="http://people.ericsson.se/gifs/logo.gif">
    </TD>


<TABLE CELLSPACING=0 BORDER=0 WIDTH=623>


<! dette bør være dynamisk>
<h1>Welcome calling party</h1>

<p>
information string presented by the subscriber dependent on A identity
</p>
<p>
You can contact me in one of the following ways:
</p>
<!here comes the different options the calling party may be presented>



<FORM METHOD="post" ACTION="http://www.stud.ifi.uio.no/~kribolst">
<INPUT TYPE="hidden" NAME="email" VALUE="-@.">
<INPUT TYPE="hidden" NAME="realname" VALUE="-">

<INPUT TYPE="submit" VALUE="Send sms         ">

<INPUT TYPE="text" NAME="subject" MAXLENGTH="600">
<BR>
```

```
<INPUT TYPE="submit" VALUE="Send e-mail      "
action= href=mailto:Kristine.Bolstad@eto.ericsson.se>


<INPUT TYPE="text" NAME="subject" MAXLENGTH="600">
<BR>
<br>
<br>
<INPUT TYPE="submit" VALUE="leave voice message      ">


<INPUT TYPE="submit" VALUE="forward to next terminal      ">

</FORM>

</TD><TD WIDTH="50"></TD><TD>


</BODY>
</HTML>
```

Appendix B

Database

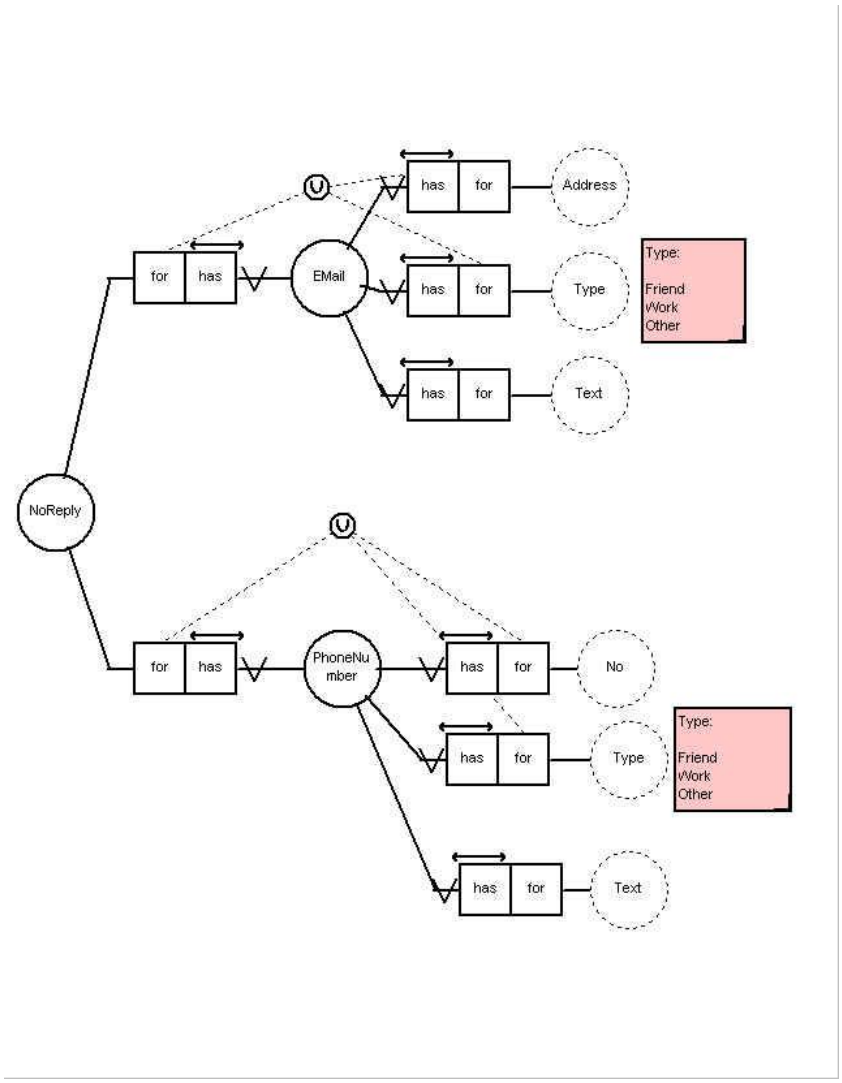
B.1 Service.xsql

```
<?xml version='1.0'?>
<?xml-stylesheet type="text/xsl" href="profiles.xsl" ?>

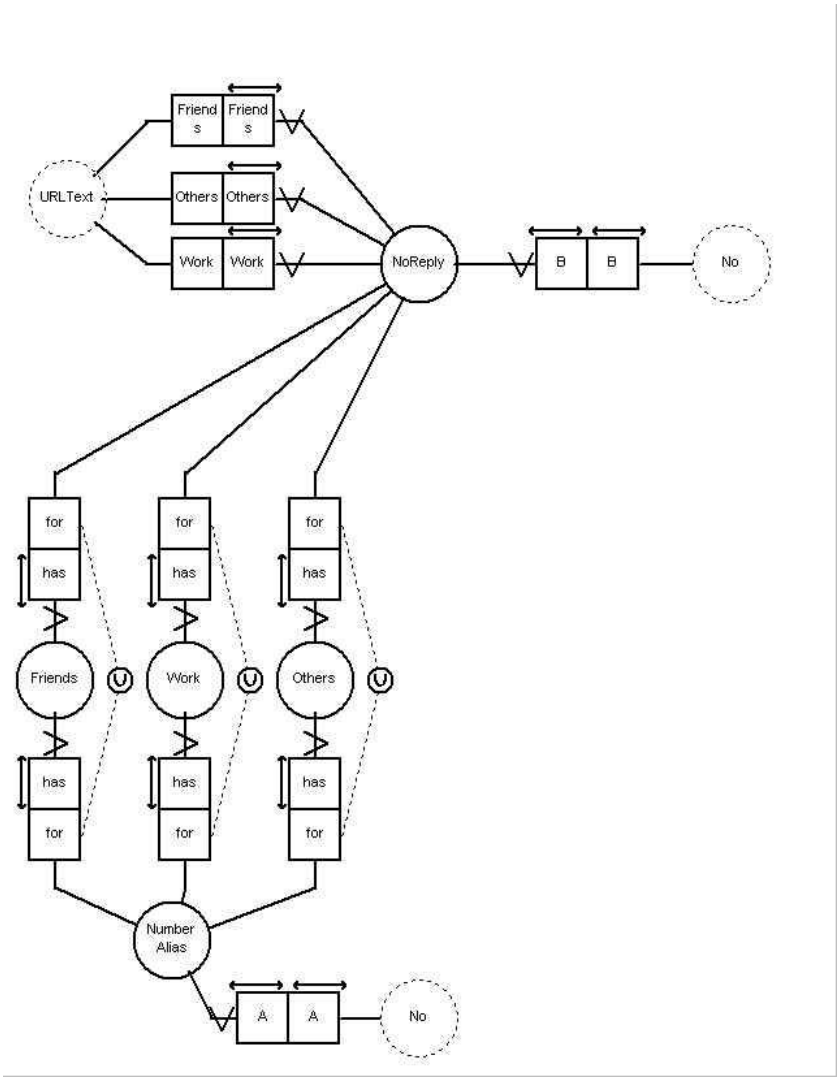
<MESSAGE OPERATION="get" connection="noreply" xmlns:xsql="urn:oracle-xsql">
        <SERVICES>
                <xsql:query NO_B="%">
                        SELECT
                                *
                        FROM
                                noreplyprofile
                        WHERE
                                NO_B LIKE '%{@NO_B}%'
                </xsql:query>
        </SERVICES>
</MESSAGE>
```

B.2 Service.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT NOREPLY (NO_B, URLTEXT_FRIENDS, URLTEXT_OTHERS,
URLTEXT_WORK?, FRIEND_NOS?, OTHER_NOS?, WORK_NOS?, PHONE_NOS?, EMAILS?)>
<!ELEMENT NO_B (#PCDATA)>
<!ELEMENT URLTEXT_FRIENDS (#PCDATA)>
<!ELEMENT URLTEXT_OTHERS (#PCDATA)>
<!ELEMENT URLTEXT_WORK (#PCDATA)>
<!ELEMENT FRIEND_NOS (NO_ENTRY*)>
<!ELEMENT OTHER_NOS (NO_ENTRY*)>
<!ELEMENT WORK_NOS (NO_ENTRY*)>
<!ELEMENT PHONE_NOS (PHONENO_ENTRY*)>
<!ELEMENT EMAILS (EMAIL_ENTRY*)>
<!ELEMENT NO_ENTRY (NO*)>
<!ELEMENT PHONENO_ENTRY (NO, TYPE, TEXT?)>
<!ELEMENT EMAIL_ENTRY (ADDRESS, TYPE, TEXT?)>
<!ELEMENT NO (#PCDATA)>
<!ELEMENT TYPE (#PCDATA)>
<!ELEMENT TEXT (#PCDATA)>
<!ELEMENT ADDRESS (#PCDATA)>
```



B.3 Niam Diagrams of the Database solution


Friend s
Friend s
URLText
Others
Others
Work
Work
NoReply
B
B
No
for
has
Friends
Work
Others
has
for
Number Alias
A
A
No

Appendix C

OnNoAnswer

C.1 OnNoAnswer plugin

```
//Title:        My First Service
//Version:
//Copyright:    Copyright (c) 1999
//Author:       Kristine Bolstad
//Company:      Ericsson

//Description:  The NoReply is a pluon in the IPT 2.0 system that
//takes the call info from the
//call setup, consults the databse, and launches a web-pages back to the caller
//when the calleed is busy on some account.

// må nå; legge inn i database, hente info, generer html.sider og launche disse

package se.ericsson.eto.mmts.serviceagent.service.plugin.annexk;

import se.ericsson.eto.mmts.serviceagent.sce.base.sar.ServiceActionRequest;
import se.ericsson.eto.mmts.serviceagent.sce.base.sar.ContinueCallRoutingSar;
import se.ericsson.eto.mmts.serviceagent.sce.base.ServicePlugin;
import se.ericsson.eto.mmts.serviceagent.sce.base.ServiceInteractionController;
import se.ericsson.eto.mmts.serviceagent.sce.base.*;
import se.ericsson.eto.mmts.toolbox.casino.*;
import se.ericsson.eto.mmts.framework.event.EventHandler;
import se.ericsson.eto.mmts.serviceagent.see.main.SaDebugGroup;
import se.ericsson.eto.mmts.framework.server.Server;

import java.awt.Frame;
import java.awt.FileDialog;
import java.io.File;
import java.io.IOException;

//import com.apple.mrj.MRJFileUtils;
```

```
import java.net.*;
import java.net.URLEncoder.*;
import java.io.*;
import java.lang.*;
import java.lang.Runtime;
import java.util.*;

import java.lang.*;
import javax.servlet.*;

public class NoReply extends ServicePlugin {

/**********************************************************************
/*                      Varaibles
/**********************************************************************


  public static final long  SerialVersionUID  = -1111111111111111L;
  public static final int   SPID              = (int)(SerialVersionUID^
  (SerialVersionUID >>32));
  private Monitor[]         mons              =
  {Monitor.CALL_RELEASE_CALLEE_INTERCEPT};

  private static String called = new String();
  private static String caller = new String();
  public static ServiceServlet serviceServlet;
  private static EventHandler     eventHandler = Server.getEventHandler();


/**********************************************************************
/*                      Constructor
/**********************************************************************

  public NoReply()
  {
    //debug
  }

/**********************************************************************
/*                      Methods
/**********************************************************************

  public int getId()
  {
    return SPID;
  }

  public static long getUID()
  {
```

```
    return SerialVersionUID;
  }


/* The invite method tells the servicepack if we want to be called in or not.
In this case we want to be called in on the INVOKE and on NO_ANSWER_INTERCEPT
events if in a terminating half call, else not.
i.e releaseComplete, busy, no answer*/

  public boolean invite(int eventId, int callDirection)
  {
   if(eventHandler.DEBUG)
      eventHandler.debug(SaDebugGroup.SERVICEDATA, 9, "OnNoAnswer",
                            "invite", "invite of OnNoAnswer plugin");
   switch(eventId){
    case ServiceInteractionController.EVENTID_INVOKE:
    case ServiceInteractionController.EVENTID_NO_ANSWER_INTERCEPT:
    case ServiceInteractionController.EVENTID_CALL_RELEASE_CALLEE_INTERCEPT:
      if (callDirection==TERM_CALL)
      return true;
      else
      return false;
   default:
      return false;
    }
  }


/*In the handle invoke we will ensure that we are notified on a no answer
event og if B is busy.
We return a ContinueCallRoutingSar. This is constructed without servicePluginID,
a CallRoutingInfo and the monitor set as parameters.
*/

  public ServiceActionRequest handleInvoke()
  {
  CallInfo callInfo = getCallInfo();
  ServiceActionRequest sar;

    //Finner B-nummeret
    called = callInfo.getDestinationAliasInfos()[0].toString();
    System.out.println(called);
    //NoReplyServlet.setCalledId(called);
    //Finner a-nummeret
    caller =  callInfo.getSourceAliasInfos()[0].toString();
    System.out.println(caller);
    //NoReplyServlet.setCallerId(caller);

    getInfomationFromDatabase(called,caller);
```

```
    //System.out.println("WIO -- User: " + getUser().getAliases()[0] + " is
    //connecting to " + callInfo.getDestinationAliasInfos()[0]);
    sar = new ContinueCallRoutingSar(SPID, new CallRoutingInfo(getCallInfo
    mons);

    return sar;
  }/*end of handleInvoke*/


/*In the handleNoAnswerIntercept method we replace the B number with
something and call on a metohod who will identify A and find his port-no
an ip-no.
We will then call on the method who vill send this url to A.
Since we do not know the IP-address and the port numer from any call setup,
is not possible at the time to load an homepage tp the calling party A.
This must be paramaters included in the call setup message, and thus the
CallRoutingInfo-object.
To overcome this I pretend to have the information.
*/

  public ServiceActionRequest handleNoAnswerIntercept()
  {
    ServiceActionRequest    sar;
    CallRoutingInfo         cri;

    cri = new CallRoutingInfo(getCallInfo());
    sar = new ContinueCallRoutingSar(SPID,cri);
    if(eventHandler.DEBUG)
      eventHandler.debug(SaDebugGroup.SERVICEDATA, 9, "NoReply",
                         "handleNoAnswerIntercept",
                         "answer-intercept of NoReply plugin");


    return sar;
  } /*end of handleNoAnswerIntercept

/*In the handleCallreleaseCalleeIntercept method we replace the B number
with something and call on a metohod who will identify A anf find his
port-no an ip-no. We will then call on the method who vill send this url to
*/

  public ServiceActionRequest handleCallReleaseCalleeIntercept
  (byte reason, byte cause)
  {

    ServiceActionRequest    sar;
    CallRoutingInfo         cri;

    cri = new CallRoutingInfo(getCallInfo());
    sar = new ContinueCallRoutingSar(SPID,cri);
```

```
  if(eventHandler.DEBUG)
    eventHandler.debug(SaDebugGroup.SERVICEDATA, 9, "OnNoAnswer",
                           "handleCallReleaseCalleeIntercept",
                           "release-intercept of NoReply plugin");

return sar;
}


// finne ut om called abbonerer på tjenesten

public static void getInfomationFromDatabase(String called, String caller)
{
String browserName = "c:/test_explorer/IEXPLORE.EXE";
URL  url;

// if B_no == A_no the subscriber wants to
//update his subscription page,
//The look up in the databse will return
//B_no.subscriptionPage
// with all lists editable for the subscriber.

if (called!=caller)
{
  // subscriber
  url = subscriberProfile.getProfile();
}
else
{
  //caller
  url = subscriberProfile.getprofile(caller);
}
serviceServlet = ServiceServlet.getOnNoAnswerServlet();
if (serviceServlet==null)
{
  // something wrong - give notification
}
else
{
  serviceServlet.handleUrl(url);
}
if(eventHandler.DEBUG)
    eventHandler.debug(SaDebugGroup.SERVICEDATA, 9, "OnNoAnswer",
                         "handleInvoke",
                         "invoke of web-page +in OnNoAnswer plugin");
  try
  {
    Runtime.getRuntime().exec(new String[] {browserName, url});
  }
```

```
    catch (IOException exc){}
    exc.printStackTrace();
    }
  }
 }
 /* end of class */
}
```

C.2 Test Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.util.*;

public class OnNoAnswerServlet extends HttpServlet {

  // doGet() will create the form
  public void doGet (HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
  {
    res.setContentType("text/html");
    ServletOutputStream out = res.getOutputStream();
    String outstr = "<HTML>";
    outstr = outstr + "<HEAD><TITLE>Friends & Family List</TITLE></HEAD>";
    outstr = outstr + "<BODY>";
    outstr = outstr + "<FORM ACTION=\"/_SERVLET_PATH_GOES_HERE/Form\"
METHOD=post>";
    outstr = outstr + "<H1>Friends^Family:</H1><P>";
    outstr = outstr + "<SELECT SIZE=5 NAME=name>";
    outstr = outstr + "<OPTION VALUE=Mamma>Mamma";
    outstr = outstr + "<OPTION VALUE=Marte>Marte";
    outstr = outstr + "<OPTION VALUE=Ann Kristine>Ann Kristin";

    outstr = outstr + "</SELECT><P>";
    outstr = outstr + "<INPUT TYPE=submit VALUE=\"Submit\">";
    outstr = outstr + "<INPUT TYPE=reset VALUE=\"Reset\">";
    outstr = outstr + "</FORM></BODY></HTML>";
    res.setHeader("Content-length", "" + outstr.length());
    out.println(outstr);
  }


  public void doPost (HttpServletRequest req, HttpServletResponse res)
       throws ServletException, IOException
  {
```

```
   // Object friend = null;
   String friend = null;

    friend = req.getParameter("name");

    res.setHeader("Content-type", "text/html");
    ServletOutputStream out = res.getOutputStream();
    String outstr = "<HTML><HEAD><TITLE>Results</TITLE></HEAD><BODY>";
    //
    res.setHeader("Content-length", "" + outstr.length());
    out.println(outstr);
  }
}
```