A Study of a Java based Framework for Telecommunications Services

A dissertation submitted in fulfilment of the requirements
for the degree of Master of Science in Computer Science
Massey University, New Zealand

David Ian Ferry, 2000

Abstract

In this report, we study some of the general issues surrounding the area of telecommunications service development including the history of telecommunications services, current service creation techniques and the network used by services. We also discuss the lack of service portability and reasons for it. The JAIN framework – a set of Java APIs for integrated networks – is introduced as an approach that elegantly addresses this. We present a survey of recent work in telecommunications services that relate to JAIN. This includes a discussion of the feature interaction problem, an overview of the Telecommunications Information Networking Architecture, in particular, its relationship with JAIN, and the rapidly advancing field of Internet Telephony. In order to demonstrate the effectiveness of the JAIN approach we present designs for two advanced services that use the JAIN framework. These services are Internet Call Waiting and Click-to-Dial. Finally, areas for future research are introduced.

Contents

List of Tables

List of Figures

Preface

Motivation

I came across the early stages of the JAIN framework during the summer break of '98. I was attempting to find a research project that was sufficiently pragmatic as to allow me to receive funding under the GRIF program offered by the Foundation for Research, Science and Technology. I was interested in JAIN as I had previously enjoyed networking papers and systems development, and the JAIN website talked about the convergence between the Internet and traditional telephony in a standard manner. It struck me that I had no accurate idea of how the telephone network functioned. Furthermore there was very little easily accessible documentation on its workings. I was also interested in *standard* based environments such as various Unix systems, and decided that JAIN was attempting to achieve "a good thing".

The work presented in this dissertation is useful in several aspects. It presents a number of telecommunications related concepts in a single logical unit that are dispersed in many publications elsewhere. It explains the existing wired-line telephone network in depth and discusses the exciting area of Internet Telephony which we have all heard so much about, but seen so little. It *demonstrates* the value of a standard based approach by designing services and, as such, the reader may understand the framework in far greater depth than by merely reading specification documents.

Acknowledgements

First of all I would like to thank my supervisor, Dr. Anand V. Raman, for his invaluable advice during this project. Because of Anand's comments I believe that I have a far greater understanding of the scientific process, and a greater respect for academia than I had previously. Anand also helped me find a more fitting balance between an algorithm and an implementation. I am also grateful for his input as this project is not in his field of research.

I'd like to thank SolNet, especially Murray McNae, for allowing me do this project. Thanks go to the Foundation for Research Science and Technology for funding. Project funding was particularly helpful, and without it I would probably not have undertaken a Masterate degree.

Next I would like to thank the other D's that made up the three D's at SolNet – David Long and Dr. David Page. Both of these guys taught me a lot during the research and input valuable ideas and provided insightful criticism. Hopefully we can be on the receiving end of more E-gratuitous errors from our software !

I'd also like to thank Paul Lyons of Massey University for several conversations that helped me to gain a fuller understanding of research, and one on the ideal telephone. The Institute support staff were also very helpful in keeping the machines running so that the post-grad students could concentrate on our work.

The Sun JAIN team (John Dekeijzer, Doug Tait and Rob Goedman) were extremely helpful with equipment and discussion during Supercomm '99 and the July '99 JAIN conference at Telcordia. Particular thanks go to Doug Tait for getting me up on a surf board for the first time. Praise the Lord ! The larger JAIN community is a great bunch of people. I would like to thank the following individuals from this community for various discussions we had: Steve Davis (Ulticom) and Matti Drissin (Ericsson Infotech) for equipment, Shmuel Kallner and Zygmunt Lozinski of IBM for discussions on both history and details of AIN deployment, Ravi Jain, Margaret Nilson, and Pualo of Telcordia for discussion at Supercomm and the July meeting, Colm Hayden and Aidan McGowan of APIoN.

Thanks go to the authors of several free software projects including: Linus Torvalds, Alan Cox and the rest of the Linux-kernel community, the OpenBSD and OpenSSH team, Brian Paul of Mesa, /.,lwn.net, and Richard Stallman. Without Richard's drive we wouldn't have any GNU, and the BSD source may not have been released. The lack of either of these would be a real pity.

Last but not least I'd like to thank my family for supporting me during this project and encouraging me, particularly when I didn't want to tidy the dissertation! I'd also like to thank both Dad and Nicola for proof reading.

Overview of the contents

Chapter 1 introduces the concept of Telecommunications services and notes the problem of service portability. The JAIN framework is introduced as providing a possible solution to the problem of service portability. The objectives of our research are also presented. Chapter 2 discusses the technologies present in the current wired-line telephone network. This includes an outline of the evolution of the wired-line telephone network, an explanation of the current day architecture named the AIN – Advanced Intelligent Network – and an overview of the protocol stack used in the AIN. The JAIN initiative is then discussed and is followed by a survey of related work. Chapter 3 introduces related areas of work in both telecommunications services and recent network architectures. Other Java-based telephony initiatives are presented and discussed. Chapter 4 illustrates architectures for two JAIN based services: Internet Call Waiting and Click-to-Dial. A summary of the work presented in this thesis and recommendations for areas of possible future work is finally presented in Chapter 5.

Chapter 1

Introduction

1.1 Telecommunication services

Telecommunications service providers (hereafter referred to as carriers) traditionally supply the necessary infrastructure to enable telephone calls to be made. As a customer typically has a choice between many possible carriers, carriers are required to differentiate themselves from competition if they are to maintain their existing customer base or expand it. A method used by carriers to attempt to achieve this differentiation is by offering *Telecommunications services or features.*[1] Once subscribed to a feature the customer receives functionality that is not delivered in a normal call. From the carrier's point of view, the offering of features also has the beneficial side effect of creating new revenue opportunities.

[1]There is a distinction between a service and a feature. In both the ITU-T – International Telecommunications Union (ITU-T, 1992) – and Bellcore (Bellcore, 1991) standards, features are portions of services that the service subscriber can distinguish. Hence a service may contain several features. The terms *service* and *feature* are used interchangeably throughout the text

Some examples of services include:

- The voice mail service. This is a service where a caller is able to leave a message in the called party's *voice mailbox.*
- The 800-number service. This service allows people to dial a number toll-free. The called party is charged for the call.
- The calling card or alternative billing service. This service allows a subscriber to charge a call to a particular account regardless of the caller's physical location.
- Time based routing. This service allows a subscriber to have a call to a particular number redirected to another number based on the current time.

As many carriers provide services it is important for carriers that they are able to create and deploy services throughout their network quickly and cost effectively in order to maintain differentiation from competitors. This requirement has led to the introduction of a number of highly effective technologies. The network that provides call setup and termination is highly fault tolerant. At present, services can be rapidly created and deployed through the use of graphical representations of both components which may be pieced together to form services, and the network which the services are deployed on. Standards exist that ensure interoperability between equipment vendors.

1.1.1 JAIN

In spite of the many desirable aspects of current service creation techniques such as rapid creation, ease of deployment and inherent fault tolerance, they

suffer from the serious drawback of non-portability. While interoperability standards exist ensuring communication between services executing on different vendors' equipment, a service that is created on vendor A's platform would have to be redeveloped to execute on vendor B's platform. This is due to the lack of standard *programming* interfaces. The JAIN program (Sun Microsystems, 1999a, 1999b) is aimed at addressing this problem and extending the service creation paradigm. JAIN is a set of open application programming interfaces (APIs) for the Java programming language (Gosling et al., 1999) that both include and extend the scope of traditional telecommunications service development. That APIs are open means that their specification is in the public domain. Futhermore, they are *standard* extensions to the Java platform. Standardization is important as any implementation of a JAIN interface must pass a compatibility test suite ensuring that it functions as anticipated. The Java platform allows software written in the Java programming language to run on any combination of operating system and hardware without modification to the software. Services can now be written in Java to use the JAIN APIs and execute on any vendor's platform which supports both the JAIN APIs and Java run-time environment.

Our work considers the above issue of service portability in some detail and studies the JAIN approach through the discussion of services that use the JAIN framework. It includes a number of objectives. These are as follows:

- A survey of work relating JAIN to existing work in the field of telecommunications.

- Identification of alternative frameworks or architectures that are similar to JAIN.

- Discussion of other Java based telephony initiatives.
- Illustration of JAIN's suitability for the development of portable services by building one or more services that demonstrate the value of a vendor and platform independent framework.

An understanding of the workings of services in the present day wired-line telephone network and telecommunications service development in general will help in appreciating the value of a JAIN based approach. This is therefore discussed in some detail in the following chapter.

Chapter 2

Background

2.1 History of the wired-line telephone network

2.1.1 The Plain Old Telephone System

Until the mid 1960's service logic was hardwired in the telephone switching systems. Network operators would meet with switch vendors and discuss the types of services customers required. They would negotiate the switching features that could provide these services and agree on a date for the feature availability. Once this was complete the network operator would plan the deployment of the service in the network switches.

Any network operator who used switches from multiple switching vendors had to repeat this process for each vendor. This was because services were hardwired and different vendors produced switches that were different internally. Therefore if a service was to be offered uniformly across the carrier's network each switch would have to be modified to support the service. The effect of this situation was that the same set of services were not able to be offered in different areas of a carrier's network. Hence a customer in

one city might be able to subscribe to service A and service B, whereas a customer in another city might only be able to subscribe to service B. Once these services were implemented they were not easily customized. The network operator would have to negotiate a change with the switch vendor. It is easy to see that this process of planning and implementing new services took several years.

2.1.2 Stored Program Control

In the mid 1960's switch vendors introduced stored program control (SPC) switches. This enabled services to be produced and modified more easily as they were programmed in software on the switch rather than hardwired. Yet, the service logic within the SPC switches was not modular. Hence it became increasingly difficult to introduce new services in the switches because the service depended upon its implementation, the service logic. This was compounded by a mix of SPC and non-SPC switches in a network. Thus services were again not offered equally across a carrier's network.

2.1.3 Traditional Call Management

The aspects of control and management of call processing is usually described using the term 'signaling'. The Plain Old Telephone System used the same data path to set the call up between the switches (whether they were SPC switches or not) as the one the voice data was carried on. That is, the control information for a call was carried in the same frequency range as the human voice. This is known as in-band signaling, where control and data information are transmitted within the same communications channel. It can be seen from this that switches could communicate only at the start

and end of a call, not while a person was using the phone line. It was also possible for a person to reproduce switch to switch communication frequency patterns, essentially reducing the integrity of the switching network. If there were multiple switches involved in the routing of a call the call setup process used a trunk line between each switch. If the dialed number was busy then each of those lines was used for the period of time required to return a busy signal. This was considered to be inefficient use of line capacity.

2.1.4 Out-Of-Band Signaling

Telephone networks overcame the problems of in-band signaling with the advent of the common channel Signaling System Seven (SS7) network. The SS7 network introduced a separate digital network used for call control or signaling activities. As the signaling takes place over a physically different network, it is categorized as out-of-band signaling. The switches would now use the SS7 network for call management and the traditional trunk as voice lines. The end user's interface (the telephone) was not required to change as the switches still recognized the numbers being dialed in-band. The network layout used at the time of SS7 introduction is shown in Figure 2.1

Out of band signaling brought the following advantages.

- A reduction in fraudulent usage of the telephone system.
- Reduced delay in call processing activities such as set up and tear down.
- Improved network reliability.
- The ability to signal during a call rather than only at the initiation and termination of a call.

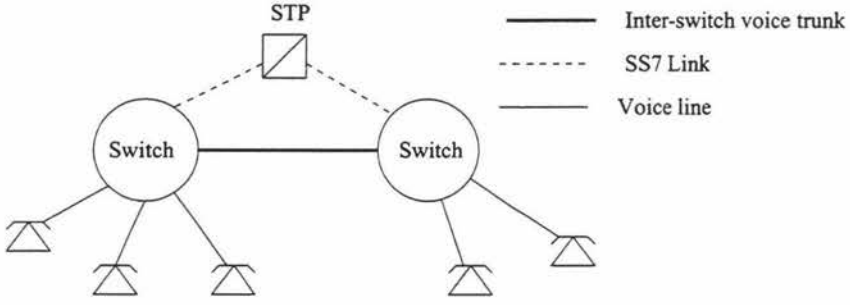

Figure 2.1: The SS7 enabled switching network. Each switch is represented by a circle. Switches were connected via two networks.The voice trunk network and the signaling network. Each node on the signaling network is assigned a unique SS7 address or *Signaling Point Code* (SPC)

The SS7 network in North America contained an extra element termed the *Service Transfer Point* (STP). The STP was introduced for the following reason: A set of nodes in a network may communicate as long as there is a physical connection between them. The most obvious case of a configuration of a set of nodes is one in which each node is connected to every other node. This is known as a *fully associative network.* Obviously a fully associative network is expensive to construct and maintain. In contrast the STP allows any node on the network to communicate with any other node on the network, as long as each node has a physical connection with one or more STPs. The STPs then route the messages appropriately. STPs provide several different routing functions which allow differing levels of redundancy in the path from one node to another, hence supplying differing levels of fault tolerance. These different paths are known as *link sets.*

The introduction of SS7 signaling allowed the introduction of a new set of services. These are collectively termed *Custom Local Area Signaling Services* (CLASS). CLASS features are Bellcore standard features that extend service

operation across switches. Some CLASS based services are presented in the following list:

- *CLASS Display Features* display information pertaining to the call on the customer premises equipment. This enables services such as *Caller name display* or *Caller number display.*

- *CLASS Security Features* enable the subscriber to block certain phone numbers. Subscribers can also activate a trace of the most recent incoming call.

- *CLASS Call Screening Features* permit the subscriber to create a list that will be used as a basis for services such as call forwarding or distinctive ringing.

- *CLASS Convenience Features* use information on the last call placed into or from a subscriber's number. This enables services such as *Automatic Recall* to quickly return a missed call or *Automatic Call-back* which lets the subscriber know when a previously busy number is free.

2.1.5 The IN/1 Architecture

The progression to SS7 based networks was a leap forward for carriers. However, carriers requested that services be developed and deployed still faster. In order to solve the problem of non-uniform service offerings and to allow carriers to produce services that were independent of switching vendors, carriers requested standard interfaces between the elements of the switching network. Bell Communications Research (Bellcore) responded to these

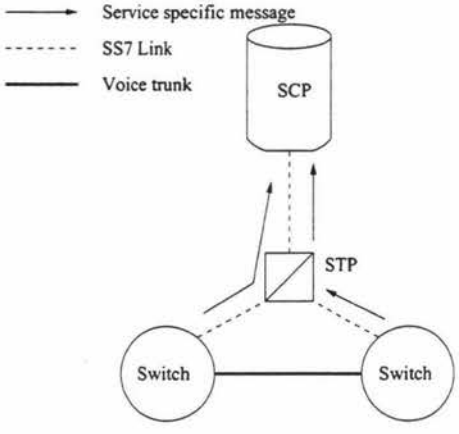

Figure 2.2: The Intelligent Network 1

requests with the concept of the Intelligent Network 1 (IN/1) in the mid 1980's. Its architecture is shown in Figure 2.2.

The IN/1 architecture was the first that placed service logic outside the switching nodes. Instead it was placed on computers with databases called Service Control Points (SCPs). The switches supported service-specific 'hooks' which would cause logic on the SCP to execute in pre-arranged circumstances. IN/1 enabled the introduction of both the 800-number service and the calling card service. However the IN/1 architecture only allowed service specific messages from the switch to the SCP. In order to communicate with the service logic on the SCPs, the switches had to be modified on a per-service basis. The modifications to the switching software essentially enabled the switch to recognize when it needed to gain information from the SCP via the SS7 network. This limitation of the IN/1 architecture led to the development of its successor - AIN.

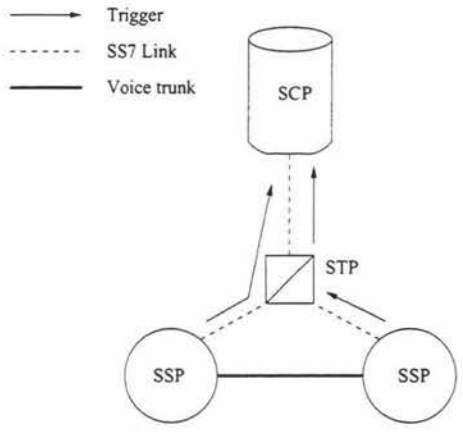

Figure 2.3: The AIN or IN architecture. This closely resembles the IN/1 architecture described earlier in Figure 2.2. However, the difference lies in the generic, standard communication between the switching system and the service control points.

2.1.6 Advanced Intelligent Network Architecture

The IN/1 architecture allowed new services to be created, but switches still had to be modified to access the SCP for these new services. SCPs were limited to running only one service. These limitations gave rise to the Advanced Intelligent Network (AIN). The AIN is also referred to as the Intelligent Network (IN) especially in European publications.

At first glance it is easy to mistake the AIN architecture for that of the earlier IN/1 architecture shown in Figure 2.2. However the difference between them is subtle and critical. This difference is not in network layout, but rather in the messages and the times that the messages occur between the switching systems and the service control points. A set of messages for generic call processing is defined in the AIN. These messages differ from the IN/1 messages in that they can be re-used across services, rather than being

unique to a service. The set of messages is collectively referred to as *triggers*. Triggers occur in predefined standard circumstances. If the switch's software contained support for triggers then many different services could be created without having to modify the switch. The AIN refers to switches as *Service Switching Points* (SSPs).

This difference between AIN and IN/1 can be illustrated using the combination of 800-number and 900-number services. In IN/1 the switch would require specific programming to notify an SCP when the first three digits dialed were '8-0-0'. Any other digits would not cause the switch to notify the SCP. However the carrier may wish to provide a 900-number service which would charge the dialer a predefined sum of money rather than the called party as is the case with 800 numbers. Under IN/1 the switch would have to be reprogrammed to handle the 900 case. When AIN is enabled any three digit or n-digit combination could be used to trigger the SCP. The only modification to the switch might be through a management interface which would request that the switch triggers if the first digits collected in a dialed number are '9-0-0'.

An 800-number service is shown in Figure 2.4. This service illustrates a use of the AIN. SSPs are conventionally denoted by circles, STPs by squares and SCPs by cylinders. Furthermore, STPs and SCPs are always deployed in redundant pairs and so some notations may also make use of shadowed symbols to denote this redundancy. The diagram also shows an example of the way in which they interact.

Message 1 represents a request from an SSP to an STP requesting 800-number translation. This request is passed on to an SCP by the STP (message 2). Message 3 is the response of this SCP to an STP. This response contains the translated number. Message 4 is the information from this

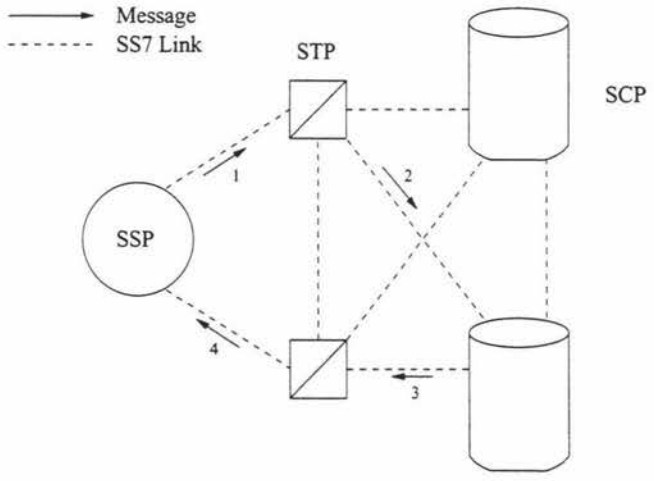

Figure 2.4: An 800-number translation service illustrating primary components of the AIN. The messages are explained in the text.

STP being passed back to the originating SSP which can then proceed to select the correct trunk. Control information between these three types of network elements is exchanged using the Signaling System 7 (SS7) protocol in a separate band from the voice data. In the next section we will discuss how a service is developed in the traditional fashion.

2.2 Telecommunications Service Development

The architecture of the AIN is such that AIN software vendors are able to pair service creation software with computer graphics, allowing developers to deal with service creation at an abstract level and doing away with the need for traditional programming. Service creation is seen as the process of linking together a number of primitives, each of which performs a single well-defined function and is capable of being represented graphically as an icon. These

primitives are called Service Independent Building Blocks or SIBs. SIBs present in Service Creation Environments (SCEs) are usually based on the specifications of Intelligent Network Capabilities set 1 and Capabilities set 2 (IN CS-1 and IN CS-2). These SIBs and their application are also known as the Intelligent Network Applications Part or INAP. IN CS-1 defines a Basic Call Process (BCP) SIB which provides basic call capabilities. This SIB has interfaces which can be used to attach additional SIBs to modify the behavior of the call. IN CS-1 defines a further 15 SIBs (ITU-T, 1995a) which can be used to customize the BCP SIB, thereby creating a service. These are described in Table 2.1.

SIB	**Description**
Algorithm	This SIB can apply a mathematical algorithm to the data passed to it.
Authenticate	Provides an authentication function to establish an authorized relationship between the service logic and a database on behalf of a user. This function can be defined by the service developer.
Charge	The charge SIB is used for resource specific charging and may be invoked several times in a service. This SIB is expected to be customized to be compatible with the carriers billing system.
Compare	Provides three logical operations, less than, greater than, or equal to. More complex logic can be created by chaining multiple instances of this SIB together. This might be used to check something such as the time of day.
Distribution	This SIB allows call logic to be distributed to its different outputs based on the service developers' algorithm. Hence it can be used to produce conditional branches in the logic.

SIB	Description
Limit	The limit SIB is intended to limit the number of calls to a particular service. This limitation may be defined by the service developer.
Log Call Information	This SIB logs detailed call information into a file.
Queue	The Queue SIB provides sequencing of calls to be placed to a destination.
Screen	Matches an appropriate identifier against a list. It can be used to verify a user ID or block particular callers.
Service Data Management	This SIB performs the appropriate actions on data stored within the network. For example, this SIB could be used to retrieve a customer's call forward number.
Status Notification	The Status Notification SIB provides inquiries about the status or changes of status of network resources.
Translate	This SIB is able to translate input information. For example it might be used to translate the dialed digits into a numbering plan which is used to route calls.
User Interaction	This SIB allows a service to send announcements and receive input to or from either the calling or called party. These announcements and the collected information may be of several forms. For example this SIB might be used to implement a voice mail service which plays an audio message to the caller asking for their PIN number. Here the announcement is an audio message and the caller's input is the sequence of digits pressed.
Verify	The verify SIB provides information confirming that the received information is syntactically correct.

Table 2.1: The SIBS in the Intelligent Network Capability Set 1

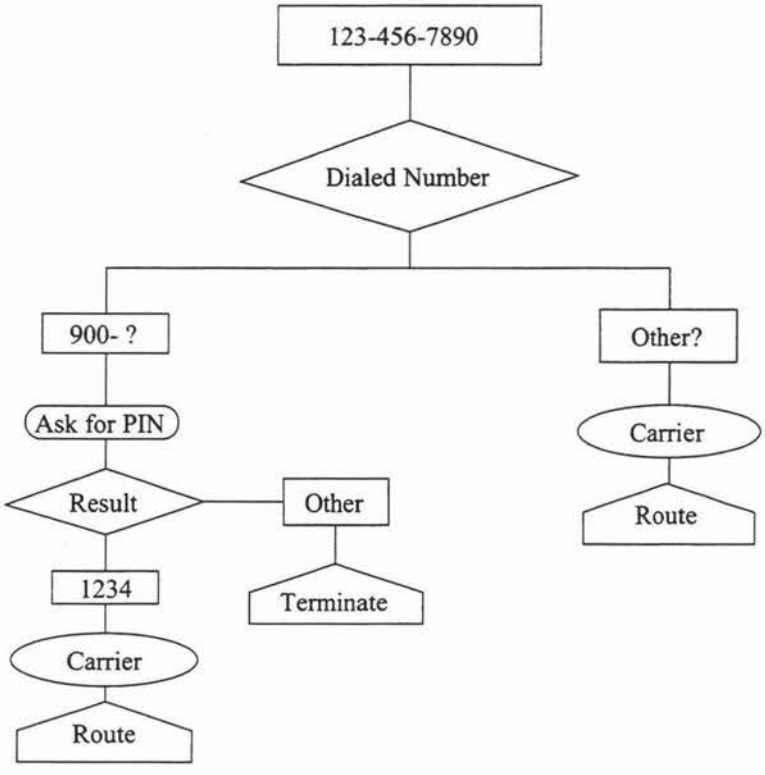

Figure 2.5: An example of the development of a Call-Screening service in AIN. Each block in this diagram represents a SIB. It is to be noted that this diagram is intended to illustrate service development, rather than show the actual SIBs used in an implementation of a service.

Figure 2.5 shows an example of a typical service creation screen for an outgoing call screening service. In this service a subscriber wishes to screen all outgoing calls for 900 numbers, yet have the ability to bypass this screening upon entering a valid PIN ("1234"). Each of the boxes in the picture represents pre-built components available in the service creation environment.

To produce this service, the developer has picked out the various icons

from a menu, arranged them in the layout representative of the service being provided and created the appropriate links between them. Once developers are thus satisfied with a graphical representation, they will input the resultant picture file to a pre-processor which will generate high-level programs in languages like C or C++. These can subsequently be compiled and installed on the nodes of the AIN. Usually these nodes are a combination of the AIN's Service Control Points (SCPs), typically computers that are running Unix, and the Service Switching Points (SSPs), which are telephone switches equipped with hardware and software enabling them to communicate with other elements.

It is easy to see how a developmental methodology such as the one outlined above can be very useful in practice. It hides from the developer many of the low-level details regarding how the service is actually implemented, reducing both the developer's cognitive load as well as the number of idiosyncrasies in the finished product. In the above example, for instance, we see that the developer only has to specify that a PIN is required at some stage in the transaction and that the PIN should match a previously submitted one. In practice, the correct PINs might be stored in encrypted form using proprietary databases such as Oracle or Ingres at the SCP. The SSP must thus request the SCP to perform this authentication and the SCP must subsequently return a result indicating the authentication outcome. These exchanges take place over a separate control network using well defined protocols. But service developers need to know none of this. They need only be concerned with the various programs generated by the picture processor and the instructions regarding which network element of the AIN each should be installed on.

Although this graphical approach has several advantages, there are some

serious shortcomings as well. For instance, standards documents which dictate the particular ways in which SIBs can interface with each other have been issued by multiple standards bodies. Furthermore they have been issued without reference implementations. To make matters worse, there are several different Service Creation Environments (SCEs) each of which has its own set of SIBs and vendor specific interfaces to them. So the likelihood is low that a picture file generated using one proprietary development environment can be successfully compiled in another development environment, let alone be installed and execute successfully. Consequently, telecommunications equipment providers (vendors) have implemented these standards to run on their proprietary hardware and provided proprietary application programming interfaces (APIs). This has the effect that it is very difficult to provide telecommunication services which can be developed on one vendor's equipment and subsequently run on another vendor's equipment without re-developing the service.

The service generated by this procedure communicates with the switching system using the SS7 protocol stack. This is discussed in further detail in the next section.

2.3 The SS7 protocol stack

Each node in a Common Channel Signaling System number 7 (SS7) network is assigned a unique SS7 address called a *Signaling Point Code* (SPC). SS7 is comprised of a suite of protocols. A loose comparison of the SS7 protocol stack and the International Standards Organization's (ISO) Open Systems Interconnect (OSI) reference model for data communications is shown in Figure 2.6. It can be seen from this diagram that the lowest three layers of

the SS7 protocol stack correspond with the physical, data-link and network layers of the OSI reference model. The TCAP and ISUP protocols are directly used by services. Hence they are included in the application layer. Each of the protocols shown in Figure 2.6 is briefly described in the following sections. These are as follows:

- Integrated Services Digital Network User Part (ISUP).
- Transactional Capabilities Application Part (TCAP).
- Signaling Connection Control Part (SSCP).
- Message Transfer Part levels 1, 2 and 3 (MTP Level 1, MTP Level 2 and MTP Level 3).
- Telephone User Part (TUP).

2.3.1 The ISDN User Part

The Integrated Services Digital Network (ISDN) User Part (ISUP) (ITU-T, 1995b; ANSI, 1995) provides call setup, termination and routing between Service Switching Points (SSPs). ISUP is used for signaling in both digital circuits (ISDN) and analog circuits (voice trunks). It is used when the destination phone number is not serviced by the local switch. For example, if two subscribers, Alice and Bob, are both serviced by the same switch (assuming they both subscribe to the same carrier and reside in the same area) then the switch does not need SS7 access to set the call up. However if Alice and Bob are not serviced by the same switch then both switches must be involved in managing the call using ISUP.

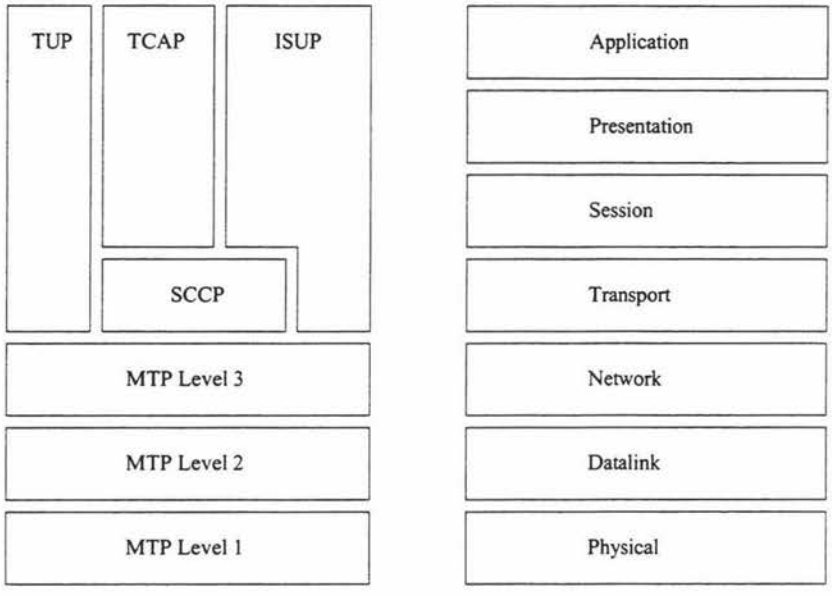

Figure 2.6: The SS7 protocol stack and the ISO's OSI reference model for data communications.

Two standards bodies have produced different definitions of ISUP, these being the American National Standards Institute (ANSI) and the International Telecommunications Union (ITU). There are also many country specific modifications to ISUP. To illustrate the use of the ISUP protocol consider the following typical situation:

> "Alice and Bob are in different cities. Alice calls Bob and Bob answers his phone. They converse for a period of time and Alice ends the phone call."

Figure 2.7 shows the ISUP messages that occur during this procedure. These messages are as follows:

1. The originating switch (Alice's switch) sends an initial address message

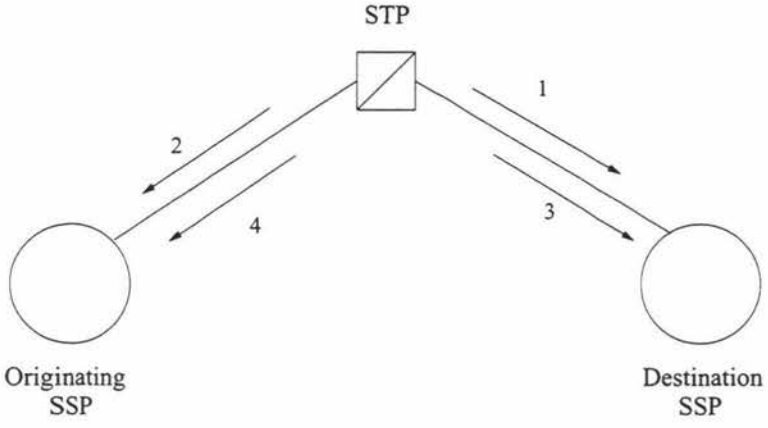

Figure 2.7: An example of call setup and termination using the ISUP protocol. An explanation of the messages is given in the text.

(IAM) to reserve an idle trunk circuit between the two switches. The destination switch (Bob's switch) receives this message and determines that Bob's line is not busy. Hence it generates a ringing signal on Bob's line.

2. Once Bob answers the call the destination switch sends an answer message (ANM) to the originating switch. Hence Alice's line stops carrying the ringing sound and her voice trunk is logically connected with Bob's.

3. When Alice hangs up the phone, the originating switch sends a release message (REL) to the destination switch.

4. The destination switch receives this message sending back a release complete message (RLC) and marks Bob's line as free.

Different ISUP specifications follow this message sequencing closely. However, the message formats and appended data vary widely.

2.3.2 Transactional Capabilities Application Part

When a switch needs extra information to place a call, the ISUP call processing is suspended and the switch uses the Transactional Capabilities Application Part (TCAP) to query one or more Service Control Points. Like the ISUP protocol, TCAP is defined by multiple standards bodies. However it is not as widely modified as ISUP and the differences between the standards are not as significant.

As the name implies, the TCAP protocol provides *transactional* capabilities. A transaction is a group of operations that are bound together to present a single success or failure indication. If a failure occurs in any of the operations, the state before the failed transaction is the same as the state afterwards. To illustrate this concept consider the transferral of $500 between two accounts. In order to accomplish this task a number of steps must be performed. At a minimum, $500 must be added to the destination account and $500 must be deducted from the source account. If an error occurred which stopped either of these steps from taking place successfully then a number of error states may exist. For example, the person requesting the transfer may receive an extra $500 if the deduction failed and the addition was successful. The person requesting the transfer may lose the sum of money if the addition failed and the deduction was successful. If the addition and deduction operations comprised a transaction then the failure of either of the operations would result in the failure of the transaction, hence the state after the failed transaction should be the same as the previous state.

A single TCAP conversation can consist of a number of transactions. The shape of each transaction is expressed by the use of *dialogues*. Dia-

logues contain a number of *components* which indicate the desired intent of the remote node. Each component can carry a number of parameters that specify extra data.

For example if an SSP required an 800-number to be translated, it might initiate a TCAP conversation with an SCP asking the SCP to translate the number on its behalf. This conversation might be shaped by two TCAP dialogue messages forming a Request-Result pair. This might be comprised of a *Begin* dialogue which contained the *Invoke* component. The invoke component would convey the switch's request to perform a task (in this case translate a number). The *parameter* of the invoke component would contain the 800-number. In turn the SCP would reply with an *End* dialogue (stating that the transaction is over) containing a *Result* component. The *parameter* of the result component would contain the translated number.

A partial list of TCAP dialogues is shown in Table 2.2. These dialogues convey a large portion of the possible transaction shapes available through the TCAP protocol. Each dialogue may contain zero or more components. Table 2.3 shows some of the TCAP components. These components are representative of the possible requests available in TCAP.

2.3.3 The Signaling Connection Control Part

The Signaling Connection Control Part (SCCP) is used as a transport for TCAP and a portion of ISUP. It offers several facilities. These are as follows:

1. Application addressing within a signaling point code.

Dialogue	Description
Begin	This dialogue signals the start of a multi-part transaction.
Continue	This dialogue returns control of the transaction to the other involved node. For instance and involved service might require a number of messages to be exchanged between the SSP and SCP. The continue dialogue would be used in this case to pass information between the nodes informing the other node that it is to provide the next dialogue.
End	This dialogue marks the end of a transaction.
Uni	This dialogue is used for simplex communication. This might be used by an SSP to inform an SCP to perform a billing task.

Table 2.2: TCAP dialogues.

Component	Description
Invoke	This component requests the remote node to perform some task. The task is specified in an optional data part.
Error	The error component indicates that an error occurred in the processing of a request. The specific error is contained in the component.
Reject	This component indicates that the associated request was rejected for some reason.
Result	The result component provides information pertaining to the outcome of an earlier request

Table 2.3: TCAP components.

2. Connection-oriented and connectionless communications.

3. Global title translation (GTT).

SCCP provides a transport layer for TCAP and portions of ISUP which can be connection-oriented or connectionless. Application identification is enabled through the use of a *sub system number (SSN)*. Each application registers one or more SSNs on a signaling node thereby uniquely identifying that application. This facility enables multiple services to exist on the same signaling node. For example a single SCP may serve both a calling card service and a 900-number service provided it has the processing capacity.

SCCP also provides a facility termed *Global Title Translation* (GTT). GTT is used to transform a unique address termed a *Global Title* into a destination point code and sub system number that together allow a service to be located. Examples of *Global Titles* include a dialed 800-number, or a calling card number. Below, we present an example of GTT.

A subscriber of a calling card service dials an access number and is prompted for their calling card number. We assume that this customer is connected to a switch owned by carrier A and their calling card has been purchased from carrier B. Now the originating switch will have to ask a Service Control Point (SCP) to verify that the card number is authentic and that it contains sufficient credit. At this point the switch does not know which SCP it should send the authentication request to. Obviously the appropriate SCP will belong to carrier B (from whom this calling card was purchased). But how does the switch know this? Prior to AIN an obvious solution to this problem would be to place a database of all calling card numbers directly into every switch. This is undesirable as switches are not intended to perform service logic. Doing so would invalidate the concept

of the AIN. Instead the switch places a global title translation request to its STP. The STP simply returns the appropriate signaling point code and sub system number. The switch uses this information to communicate with the appropriate SCP. One of the roles of the STP is to provide access to the global title database. It should be noted that an individual STP is not required to store the entire set of global titles and their resolution. Instead GTT allows the global title information to be hierarchically distributed. The GTT function also enables STPs to load balance requests to SCPs. Suppose that several global titles are frequently accessed and that each of these has a set of suitable signaling point codes (SPCs) and sub system number (SSNs) pairings. The STP may then choose one of these using some algorithm to ensure that all of these SPC-SSN pairs are accessed with equal frequency.

2.3.4 The Message Transfer Part

All of the SS7 protocols mentioned above use the message transfer part to provide end-to-end communication. The message transfer part is broken into three layers named MTP Level 3, Level 2 and Level 1. MTP Level 1 usually provides the physical layer. MTP Levels 3 and 2 are often collectively referred to as the message transfer part as they can be placed over any physical medium deemed appropriate.

MTP Level 1 defines the physical interfaces used for the signaling link. These include:

- *E-1* (2048 kilo bits per second (kbps), supporting 32 64kbps channels).
- *DS-1* (1554 kilo bits per second providing 24 64kbps channels).
- *V.35* (One 64kbps channel).

- *DS-0* (One 64kbps channel).

- *DS-0A* (One 56kbps channel).

It can be seen from this list that MTP Level 1 interfaces are often specified in terms of channels. The use of channels ensures a minimum quality of service throughout the network. MTP Level 1 provides functionality similar to the ISO OSI physical layer.

The MTP Level 2 protocol uses message sequencing, flow control and error checking techniques to provide reliable end-to-end message transmission. MTP Level 2 messages must be delivered within a bounded period of time. This ensures that the signaling network provides a guaranteed quality of service. The OSI ISO data link layer and the MTP Level 2 protocol perform similar functions.

The highest level of the MTP is MTP Level 3. This protocol provides routing between the signaling nodes in the network. The routing function in MTP Level 3 provides congestion management and is capable of routing traffic away from failed network nodes. Hence MTP Level 3 and the ISO OSI network layer are comparable.

2.3.5 The Telephone User Part

The Telephone User Part (TUP) is a protocol that enables call setup and tear down between switches. It is more limited in scope than ISUP as it only handles analogue circuits and for this reason it is not widely used. China and Brazil are among the few countries that use this protocol.

We have now discussed the workings of the wired-line telephone network, service creation and the protocol stack used to access AIN services. These

topics provide us with the background necessary for studying the JAIN approach to service development.

2.4 Java APIs for Integrated Networks

2.4.1 Introduction to JAIN

JAIN (Sun Microsystems, 1999a, 1999b) is a set of Java application programming interfaces for integrated networking. The name refers to the integration between telephony related protocols, whether they are IP based or SS7 based. JAIN is managed under Sun Microsystem's Community process which allows many companies to participate in creating the various Java APIs.

The central idea behind JAIN is to provide *open* Java programming interfaces lowering the cost of developing telephony services. These interfaces are *standard* extensions to the Java platform. As the service developers will be using Java the resultant services will be able to run on any CPU/OS combination with a supported Java runtime environment. The term *open* means that the specification of the programming interfaces is available to the public free of charge.

2.4.2 The layered model of JAIN

JAIN allows the service developer to utilize the integration of IP and SS7 based networks through the use of a layered software model. This software model is comprised of three levels. The lowest level is the protocol layer which defines interfaces for telephony related protocols. Interfaces are being defined for both SS7 and IP protocols. This enables a service to issue proto-

col commands. The second layer is the call control and transactional layer. This provides a generic call control model that allows a service to express high level call related concepts. The third portion is the security interface. This allows secure, authenticated remote access to the call control layer.

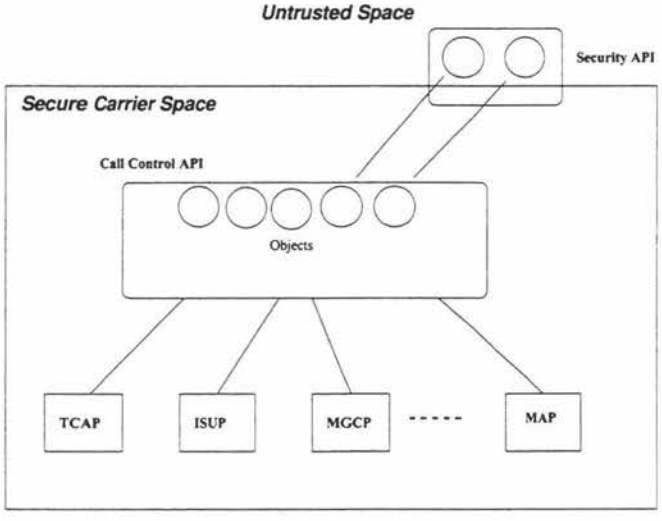

Figure 2.8: The layered model of the JAIN approach. At the lowest level are the protocols. Above them is the call control and transactional layer. Both of these are within the secure carrier's network which can be accessed remotely via the security layer.

This software model is shown in Figure 2.8. It can be seen from this diagram that both the protocol and the call control layers are inside the carrier's network. The security layer bridges the trusted and untrusted networks enabling a service located outside the carrier's network to place calls.

2.4.3 The JAIN protocol interfaces

The protocol interfaces define open APIs for SS7 and IP based protocols. Currently the SS7 protocols for which JAIN APIs are being specified include

ISUP, TCAP and MAP - the Mobile Applications Part.[1] The IP protocols for which JAIN APIs are being specified include SIP, MGCP and H.323. These protocol interfaces may be used by service developers. For example an 800-number translation service may be developed using the TCAP API by receiving the appropriate *triggers* from an SSP.

Protocol stack vendors are not required to re-implement their stacks using the Java programming language. There are a number of mechanisms which can be used to provide communication between a Java process and a machine language process. These techniques include the Java Native Interface (Sun Microsystems, 1997), use of a socket based protocol or CORBA.[2] Hence stack vendors are only required to implement the appropriate objects that communicate with the stack possibly using one of the techniques mentioned. The protocol layer may be directly accessed by a service. However, a service developer may wish to develop a service at a higher level of abstraction. JAIN's call control model supports this as we discuss shortly.

2.4.4 The Call Control layer

This portion of the JAIN model allows the service developer to express high level call related concepts such as: *'set up a call'* or *'add a party to the call'*. Hence a call may be initiated without the service developer having to understand a signaling protocol in detail. This is intended to enable faster development of telecommunications services as less time is spent programming protocol details. A desirable side effect of this model might include a

[1] If the JAIN framework is to be used for a large array of telephony services it must provide support for mobile communications. Currently JAIN intends to support MAP.

[2] CORBA - Common Object Request Broker Architecture - allows software written in different languages to communicate.

larger pool of service developers a barrier to entry is reduced.[3]

A further advantage of a generic call control model is that a service gains independence from any particular signaling protocol. An implementation of the call control API might use a number of signaling protocols. Hence the service may execute in the same fashion across a number of networks.[4] This is the utilization of integrated networks that is referred to by the JAIN program. Services that use the call control layer execute in the carrier's network, hence only carriers may offer services using the call control layer. JAIN enables third party developers to access a carrier's call control layer remotely through the security layer which is discussed in the next section.

2.4.5 The Security layer

JAIN's security layer provides a mechanism that allows controlled access from a remote service to a carrier's call control network. This enables a party other than a carrier to offer a number of services. For example, a service provider might decide to introduce a service that monitors stock levels on a particular market and contacts the subscriber via any of a number of mechanisms initiated by the stock's performance. If one of the mechanisms included notification by a telephone call, setup by the service, between the stock broker and the subscriber, for example, only a carrier would have been able to offer this service. However, the use of the security layer would allow a third party to offer this service and use the carrier's network for call control. The security layer used in JAIN is a set of Java interfaces for an existing

[3]We assume that the knowledge required to produce a service is a significant barrier to entry.

[4]If for example the ISUP, MGCP and SIP protocols were supported by the call control API implementation then a service using this API would be able to function in the same fashion on dissimilar networks.

specification called Parlay which is described in the next section.

2.4.6 Parlay

The Parlay specification (The Parlay Group, 1999) allows secure remote access to a carrier's signaling network. All data between a Parlay user (assumed to be a third party) and a Parlay provider (assumed to be a carrier) is encrypted so that it cannot be viewed by unauthorized parties without significant computation taking place on the encrypted data. Parlay users are authenticated by the Parlay framework. Hence the carrier is aware of the identity of the Parlay user. Encryption algorithms are negotiated as part of the framework.

Parlay contains a call model which allows the abstract states of a call to be used and viewed by service developers without having to explicitly encode the necessary steps to set up calls using different protocols. The specification of Parlay is technology and language independent.

A JAIN service may use any combination of components present in the JAIN model during its execution. The execution of a JAIN service may be supported by the Service Logic Execution Environment. This environment is described in the following section.

2.4.7 The Service Logic Execution Environment

It is desirable for services to have a high degree of reliablility in their execution. If a service exists on a single computing node and that node fails then the service also fails. Hence a service must execute across a number of computing nodes in order to be highly reliable. The Service Logic Execution Environment (SLEE) aims at providing such an environment that contains

executing services. Services that run within such an environment and make use of the environment's facilities are able to survive the failure of some of the nodes used to support the environment. The SLEE specification is a JAIN API that provides a specification of the facilities used by a service. Hence the service is likely to be written with respect to the SLEE API. In doing so the service itself does not have to provide load balancing and *fail over*.[5] These are received from the SLEE when a service uses the SLEE API appropriately.

The management of a carrier's network elements is perhaps as important as the development and introduction of new services. Management applications have the same portability constraints as services. The management of network elements is briefly discussed in the following section.

2.4.8 Operations Administration and Maintenance

Operations Administration and Maintenance (OAM) allows applications to send and receive information to or from telecommunications equipment and software. This information is commonly used by a management application to control a device or network stack. The current release of OAM for JAIN is the JAIN SS7 OAM API (Sun Microsystems, 1999d). It allows a management application to receive alarms and configure various levels of an SS7 stack. These are MTP Levels 2 and 3, SCCP and TCAP. Future releases of JAIN OAM may include support for IP based protocols.

JAIN is comprised of a set of APIs. These APIs may be used for network management, service development and service execution. A service that is developed in Java and uses JAIN APIs is able to execute without

[5]Fail over is the process of passing tasks from one node to another in the event of failure of the first node.

modification on any CPU/OS combination that supports the Java Runtime Environment (JRE) and provides a compliant JAIN API implementation.

There are several areas of ongoing research that are related to JAIN. These areas include a study of a the feature and service interaction problem in service development and provisioning, the evaluation of similiar approaches to the JAIN framework, and the development of Internet Telephony. Each of these is discussed in turn in the next chapter.

Chapter 3

Related and Previous work

This chapter presents a review and discussion of some of the areas of ongoing research within the field of telecommunications services that are of relevance to JAIN. These include:

- The problem of feature or service interaction.
- The Telecommunications Information Networking Architecture (TINA).
- The field of Internet Telephony.
- The integration of AIN and Internet Telephony.

We now take a more detailed look at each of the above in turn.

3.1 Feature or Service Interaction

As network operators continue to offer additional services, new services may interfere with other services causing unanticipated outcomes, such as adversely affected performance. This problem is termed *feature interaction.* Indeed, any deviation from the expected behavior of a service is deemed a

feature interaction. This problem is considered to be a major obstacle in the addition of new services to a carrier's network.

A feature differs from a service in that a feature is defined as a *"unit of one or more telecommunications or telecommunications management based capabilities a network provides to a user"* (Bellcore, 1991). ITU-T defines the term *service feature* to be *"the smallest part of a service which can be perceived by a service user"* (ITU-T, 1992). Both the service interaction problem and feature interaction problem are often referred to as feature interaction, presumably because feature interaction is the most generic term for the problem domain. Keck and Kuehn (1998) claim:

> "most interactions between service or supplementary services can be tracked down to interactions between single service features, and because an interference is defined as an undesired interaction, i.e., a special case of an interaction".

They present a survey of the field of feature interaction providing both a summary and categorical analysis of current approaches, and a perspective on the problem termed the *emergence level view.* This view acknowledges that the cause of interactions may be of many different kinds. A framework of four different criteria is presented which is used to classify different approaches to the feature interaction problem. This framework refines the existing classifications into the detection, resolution and prevention categories. The results of this classification are then complemented by the method used, the point in service life-cycle that the approach applies, and the network context of the approach. The classification framework is deemed useful due to the majority of the different approaches that fit naturally into it. In fact,

> "In the detection category, there is an overwhelming number of

formal method approaches, using all kinds of methods, while there is very little support for real implementations, especially for dealing with legacy services. In the resolution area, there are promising approaches, but many of them rely on explicit mutual knowledge of both services, which is clearly a problem in a deregulated, multi-provider environment. The prevention approaches are long term solutions that require a suitable network infrastructure. Here almost no short term perspective is seen, especially if legacy services are to be covered. Management papers are still of modest number, although the complexity and size of the problem makes mastering this area a crucial issue." (Keck & Kuehn, 1998).

Feature interaction is highly relevant to JAIN. This is because JAIN is intended to enable the rapid development of *portable* innovative services. As JAIN services may be introduced into carriers' networks it is vital that JAIN acknowledges the problem of feature interaction and attempts to integrate appropriate techniques in the field. However, the recommendation of particular techniques that should be applied in the JAIN framework is beyond the scope of our work. The problem of feature interaction extends to all service development. It is also worth noting that JAIN is not the only recent framework for the development of telecommunications services. We now discuss an approach similar to JAIN. This is the Telecommunications Information Networking Architecture (TINA).

3.2 TINA

The Telecommunications Information Networking Architecture (TINA) intends to provide a communications infrastructure that allows services to be made network independent. This is enabled through three core concepts. These are as follows:

- Computing architecture: This defines a distributed processing environment (DPE) which allows software components of services to execute in physically different locations. This environment is based on the CORBA - Common Object Request Broker Architecture (OMG, 1997).[1] The CORBA specification is modified to meet telecommunications requirements.

- Service Architecture: The service architecture defines a set of principles for providing services. It introduces the notion of a session to provide a coherent view of the varying relationships that can exist during the provision of services.

- Network Architecture: This provides a description of a generic, technology independent model for setting up connections and managing networks.

TINA includes a number of active areas of research. The areas of research in TINA presented in this survey are those relevant to JAIN. These areas include:

- Conformance testing of CORBA components.

[1] CORBA allows applications to communicate with one another wherever they are located, independently of both the programming language used to describe them and the Operating System they execute on.

- Mobile telephony use.
- Quality of service.
- Internet Telephony and TINA.

The first area we discuss is the conformance testing of CORBA components.

3.2.1 Conformance testing

In a distributed component based environment such as TINA, it is vital that any components that may be introduced into the environment conform strictly to specification. Schieferdecker et al. (1998) has carried out research aimed at ensuring the conformance of CORBA components in general. This may, of course, be applied to TINA as it uses CORBA. The work considers the conformance testing of CORBA service components in the realm of the ITU-T conformance testing methodology TTCN - Tree and Tabular Combined Notation (ISO/IEC, 1997). TTCN is the only standardized notion for protocol conformance testing. It discusses the use of TTCN for testing the computational objects of service components and proposes an implementation approach to derive executable tests from TTCN that can be executed on any CORBA compliant ORB - Object Request Broker - (with minor modifications for ORB specific adaption) to check the functional correctness of deployed service components in distributed object environments.[2] The TTCN/CORBA gateway presented by Schieferdecker et al. (1998) is a

[2]An ORB is a software component that establishes relationships between objects. By using an ORB an object can transparently invoke a method on another object which can be situated on either the same host or another host on the network.

general approach for testing distributed systems. TINA is used to illustrate the TTCN/CORBA gateway.

Schieferdecker et al. (1998) state that exhaustive testing is not practical and generally too expensive. However, they propose that each interface of a service component is tested separately. JAIN ensures conformance through a different technique. Conformance of JAIN components is ensured by the use of a compatibility test suite for each API within the JAIN framework. In this fashion the programs within the compatibility test suite perform tests on the API implementation noting when the implementation under test deviates from expected behaviour. Both of the approaches perform black box testing. However, the compatibility test suite approach used under JAIN is only of limited use as it is non-conformant with a standard for testing, unlike the TTCN/CORBA gateway mentioned earlier. Hence a particular test suite under JAIN may be more rigorous than another. The application of TTCN to JAIN is thus a potential area for future research.

As TINA uses a DPE – Distributed Processing Environment – to provide telephony services including basic call processing, bounded call setup times need to be reproduced in the environment. The next section discusses a successful approach to implementing Quality of Service in the DPE.

3.2.2 Quality of service

The computing architecture of TINA is non-trivial to implement. It requires a DPE that allows services written in any language to communicate with any other service or component within the entire TINA. As the environment is based on the CORBA specification, work has been carried out that is aimed at producing an ORB that is capable of providing the real-time

characteristics necessary for call processing. Stefani et al. (1998) present ReTINA, an approach for distributed object based middleware. ReTINA is based on a minimal but highly flexible framework for the construction of ORBs. This framework supports the introduction of arbitrary binding mechanisms including arbitrary communication protocols and stacks, as well as fine-grained control of system level resources. It allows the development of highly scaleable ORBs that, in turn, support the development of applications with temporal quality of service requirements. The ReTINA work has produced some desirable results that allow the ORB to display real-time properties. This is of great importance as there are real-time constraints on call setup times that need to be addressed in a distributed object environment if these objects are to provide call processing. JAIN provides a different approach towards call management. Instead of using distributed object processing it places open APIs above the particular signaling protocols. The signaling protocols themselves are responsible for performing call processing functionality within the required time. The work presented by Stefani et al. (1998) is applicable for any JAIN services that use distributed processing. Java's Remote Method Invocation (RMI) implementation might benefit from the framework presented by them. However, the application of these findings to RMI is beyond the of the scope of our work.

Obviously any framework for telephony will have to cater for mobile telephone use. Hence, we now discuss the support of mobile telephony within TINA.

3.2.3 Mobile telephony use

Mobile users may move between networks. It is desirable that they are able to receive the same services independent of the particular network they are using. For example, if a cellular subscriber moves between countries it is possible that the equipment they are accessing is owned not by a carrier that they are subscribed to. Thus the concept of the *home* and *visited* carrier is common in cellular networks. As TINA is intended to provide a paradigm for telecommunications in general it must accomodate the requirements of mobile telephony use. Hence mobile support in the TINA is an area of active research. We discuss three recent techniques that facilitate mobile telephony under TINA. In the TINA model, a user is represented by a *user agent* that is under the domain of the carrier the user is subscribed to. All three approaches make some modification to the way in which the *user agent* is accessed.

Lombardo et al. (1999) analyses the benefits of user agent migration to the visited carrier versus maintainance of the user agent in the home domain with respect to two criteria, the signaling load and average service response time. The results presented are gained from mathematical descriptions of the signaling load and average response time.

An alternative technique is presented by Kuepper (1999). This approach uses Mobile Agents.[3] The Mobile Agents are sent to the visited carrier and server on behalf of a user agent. This presents the problem of locating mobile TINA user agents. Three stratagies are presented by Kuepper (1999) and the response times of each are statistically evaluated and discussed.

Stamoulis et al. (1999) presents the concept of using Mobile Agents to

[3]Mobile Agents are software components that can move from host to host on a network.

select the best service offer available in the area where the user is currently located. An algorithm for the selection of the best service offer available by the agent is discussed as also are methods to determine whether the presented strategy is more effective than using fixed user agents. Each of these approaches appears to be promising. The first two strategies present performance characteristics that are mathematically derived. The agent based service selection does not present performance characteristics. However, it does discuss criteria that can be used to determine whether it is a more effective strategy than using fixed user agents.

The approaches aimed at supporting mobile telephony use in TINA are markedly different from the approach used in JAIN. JAIN intends to produce APIs that will allow service developers to access various protocols that are used by mobile services.[4] The TINA approach is drastically different as under TINA there is no distinction between basic call processing and supplementary services. Hence supporting mobile users in TINA requires modification of the user agent.

Another area within TINA that is under active research is the integration of TINA and the Internet. This is discussed in the next section.

3.2.4 Internet Telephony and TINA

As the Internet continues to have an increasingly large effect on telecommunications, the connection oriented view which TINA holds may not fit the Internet model. The TINA model has been criticized for its inability to deal with the Internet as a communications medium. McKinney et al.

[4] A particular protocol used by cellular phone is the Short Messaging Service (SMS). This enables a cellular subscriber to send and receive textual messages from and to their cellular phone.

(1998) states:

> "TINA meets many current and future telecom requirements. Nonetheless, while it set out on a very advanced track (use of object orientation, language independent interfaces, and the DPE), it seems to have ignored recent developments. Moreover, TINA started out as switching-centric and has not addressed until recently connectionless network environments like the IP. The rapidly growing Internet has fostered an environment in which new developments are immediately released to a large user base. This environment creates an evolutionary approach in which some innovations rapidly become commonplace (like the World Wide Web) while others disappear as fast as they surface. TINA still has not yet been deployed in an actual Internet user setting. On the other hand, TINA's principles provide a flexible way of deploying telecom and other advanced services. TINA offers a means to blend telephony and data services into a single service network. Basic TINA principles guide telephony and data into an approach from which both may benefit."

Recommendations pertaining to the integration of TINA and the Internet have been presented by Lewis and Tiropanis (1998). Their work examines the suitability of TINA for use in an open services market based on Internet technology. An implementation of a composite, IP-based service that uses features of the TINA service architecture is presented. The implications for the wider integration of TINA and the Internet in the delivery of open services is discussed. The work of Lewis and Tiropanis (1998) presents several insights which are of interest in the integration of TINA and the

Internet. They list them as follows:

1. The Internet already provides a wide range of service location facilities through advertising and the Web. It is not clear therefore that a specific Broker role will be necessary.

2. The delegation of service usage interactions to inter-domain reference points not covered by direct contractual business relationships may be more significant than currently indicated in the TINA business model.

3. The centrally controlled connection oriented network model adopted in TINA fits poorly with the Internet communications model. The design of TINA therefore needs to accommodate the delegation of control of the network from a centralized mechanism to the user application driven one.

4. TINA needs to recognize the persistence of signaling for QOS requests, and a systematic way of integrating such protocol based mechanisms with the DPE approach is required. The adoption of web browser as the basis of the user application should support the integration of Java based CORBA components with protocol specific components that is needed as part of such an approach.

TINA's aim of providing network independent services is highly desirable as services that are network independent are more likely to survive changes in networking technology than those tightly coupled to an underlying network. However, we believe that the approach taken in defining TINA is troublesome. The entire environment is being defined rather than the fragments required to provide such an environment. This will result in a model

that is far more complex than it needs to be. This is also pointed out in other recent work which criticizes TINA in its specification of areas that already exist in the Internet (Lewis & Tiropanis, 1998) and its slowness in evolving to maintain relevance in the advancing field of Internet Telephony (McKinney et al., 1998). JAIN also aims at producing services that are independent of an access network. However, the JAIN program enables network independent services through the use of layered programming interfaces. This is a far less complicated model than that of TINA.

The next topic in this survey looks at the rapidly advancing field of Internet Telephony.

3.3 Internet Telephony

3.3.1 Introduction

Over the last few years, Internet Telephony has grown from a computer enthusiast's hobby into a suite of technologies that are close to capable of replacing the existing PSTN – Public Services Telephone Network – infrastructure. Carriers are considering the use of Internet Telephony as a part of their infrastructure due to its cost effectiveness.[5] Because of this, a significant portion of the research carried out in Internet Telephony is related to its integration with conventional telephony. This has led to a number of architectures and technologies. Other areas of research in Internet Telepho-

[5]Internet Telephony is deemed to use bandwidth more efficiently than conventional telephony as the network which carries the voice data in a conventional PSTN typically reserves a line for use during the entire call even though both people involved in the call may be silent at times. In a packet-switched network bandwidth is only used when data travels over the network. Furthermore, Internet Telephony access devices are intelligent. This enables them to to use advanced compression techniques that reduce the data required to reproduce a voice signal.

ny include signaling protocols and methods for describing facilities such as a call processing language. The field of Internet Telephony is of relevance to JAIN as JAIN services are intended to operate across different signaling protocols.

3.3.2 Interworking IP and conventional telephony

It is generally accepted that any carrier scale IP based network will be required to operate in conjunction with the existing carrier networks. IP based devices that offer telephony functionality are expected to access the same set of *value added* services that are offered to customers today. This requirement has driven research in areas related to this problem. A number of solutions are available for the integration of the PSTN and the Internet. As different approaches are implemented by carriers, an interworking between them is needed. This is likely to be a new area of research.

The use of Internet telephony within different carriers' network types has been discussed by Kozik et al. (1998). The work evolves a number of current carrier networks, including AIN, Inter-exchange carriers, cable telephony and ISP networks to take advantage of IP based networks.[6] The evolution of each of these network types is shown in a number of steps. Innovative services are made possible in a number of cases.

3.3.3 Internet Telephony Protocols

The problem of developing mechanisms to integrate IP users with the PSTN is an area of active research. Huitema et al. (1999) focus on calls between the Internet and the PSTN while Schulzrinne and Rosenberg (1999) discuss

[6]Kozik et al. (1998) refers to IP networks by the more generic term *data network*.

techniques that are equally applicable in the case of pure Internet Telephony and Internet/PSTN telephony. Both techniques rely heavily on signaling protocols that operate across the Internet.[7]

In this section we give an overview of a number of Internet Telephony protocols and uses that are relevant to JAIN. We refer the reader to (Schulzrinne & Rosenberg, 1999) for an overview of the IETF – Internet Engineering Task Force – Internet Telephony architecture and a summary of the following protocols :

- The Session Initiation Protocol (SIP)
- The Session Description Protocol (SDP)
- The Real Time Transport Protocol (RTP)
- The Real Time Streaming Protocol (RTSP)
- The Gateway Location Protocol

While Schulzrinne and Rosenberg (1999) give a useful overview of how the IETF protocols fit together they do not cover a number of protocols that are used in Internet Telephony to ensure quality of service or non-IETF protocols. A protocol that does ensure quality of service is the RSVP – Resource Reservation Protocol (Braden et al., 1997). Non-IETF protocols that are used in Internet Telephony include MGCP – the Media Gateway Control Protocol (Cuervo et al., 2000), and H.323 (ITU-T, 1997a).

[7] www.dialpad.com is an example of a service that allows an Internet user to call a PSTN number.

3.3.4 Call Control protocols

H.323, SIP and MGCP are all used for call control or signaling. SIP and H.323 may both be used for call setup between two Internet users. MGCP and H.323 are both used for controlling PSTN/IP gateway devices. SIP may also be used for this purpose. However, at the time of writing SIP has recently reached the status of an Internet RFC.[8] Call control or signaling protocols are supported by the JAIN framework by the protocol layer and are abstracted by the call control layer. The Click-to-Dial service presented in chapter 4 may use any of these signaling protocols.

H.323

The ITU H.323 protocol (ITU-T, 1997a) is defined as a part of the International Telecommunications Union H series documents for Audiovisual and multimedia systems.[9] This series defines both protocols and procedures for multimedia communications on packet based networks including the Internet and it includes:

- H.245 for call control.
- H.225.0 for connection establishment.
- H.332 for large conferences.
- H.450.1, H.450.2, H.450.2 for supplementary services.
- H.235 for security and

[8] An Internet RFC is the public form of a protocol developed by the IETF.

[9] ITU's audiovisual and multimedia systems recommendations are available at http://www.itu.int/itudoc/itu-t/rec/h/index.html

- H.246 for interoperability with circuit-switched services.

In addition to using these, H.323 is based heavily on the ITU multimedia recommendations which preceded it. This includes:

- H.320 for ISDN.
- H.321 for B-ISDN.
- H.324 for GSTN terminals.

The encoding mechanisms, protocol fields and basic operation of H.323 are simplified versions of the Q.931 ISDN signaling protocol. H.323 uses binary encoded messages based on the ASN.1 – abstract syntax notation-1 (ITU-T, 1997b) – with PER – packed encoding rules – for ASN.1.[10]

H.323 was initially intended for multimedia communications on a LAN, and has since been modified to support additional requirements of Internet Telephony. H.323 is now in version 2. H.323 uses RTP – the Real Time Transport Protocol – to send the packetized audio/video streams.

Session Initiation Protocol

The SIP – Session Initiation Protocol (Handley, Schulzrinne, Schooler, & Rosenberg, 1999) is intended to provide call signaling on the Internet. It is modeled after both the simple mail transfer protocol (SMTP) and hypertext transfer protocol (HTTP). As with both of these protocols SIP is text based.[11] But unlike HTTP and SMTP, SIP can use either TCP or UDP as its transport as it provides its own reliability mechanism. SIP does not

[10] ASN.1 allows several encoding rules including the common BER – Basic Encoding Rules. However H.323 protocol messages use the Packed Encoding Rules.

[11] SIP's protocol messages are passed in ASCII text.

define the type of session that is established. Hence SIP can be used to establish a voice session as easily as it could establish a shared application session. *Requests* are generated by the sending entity (the client) and sent to the receiving entity (the server). Interaction with a user of SIP is managed by software termed the *user agent.* A user agent contains both a user agent client (responsible for initiating calls) and a user agent server (responsible for answering calls). There are three non user agent servers that enable a SIP conversation. These are termed:

- The *Registration server.* This server receives updates on the current locations of users.

- The *Proxy server.* This receives requests and forwards them onto another server (which is termed the *next-hop server*) that has more precise information about the location of a caller.

- The *Redirect server.* This server also receives receives requests and determines the *next-hop server* but instead of sending the requests to the *next-hop server* it sends its IP address to the client.

The proxy and redirect servers are reponsible for finding the set of servers that a call needs to traverse. This is also called *call routing.* A client in SIP requests a server to perform a task. The particular tasks that a server performs are termed *methods.* The name given to a client's request is *invoke,* that is, a client invokes a method on a server.[12] The methods used in SIP are described in table 3.1.

[12]The terms *invoke* and *method* are borrowed from HTTP.

Method	Description
INVITE	Invites a user to join a call
BYE	This closes the connection between two users in a call
ACK	This method is used for reliable message exchanges between a client and a server
OPTIONS	This offers information about capabilities. It does not set up a call
CANCEL	Terminates a search for a user
REGISTER	Conveys information about a user's location to a SIP registration server

Table 3.1: The methods of SIP.

Consider the example of a SIP session where one user, say Alice, calls another, Bob. Bob's SIP URL is sip:bob@company.com.[13] An illustration of the message exchange required to initiate this session is presented in Figure 3.1. It shows that a number of messages must be exchanged to set up a session. Alice's SIP user agent creates an *INVITE* request for sip:bob@company.com. This request (message 1) is forwarded to a proxy on the local network. The proxy resolves the host-name of *company.com* receiving its IP address. Once it has resolved the destination IP address it passes the request onto that host (message 2). The server for *company.com* knows that this user is currently logged in as bob@sales.company.com. Hence the server informs the proxy to use the address *sales.company.com* (message 3). The local proxy resolves the host-name of sales.company.com. Once it knows the destination IP it proxies the request there (message 4). The sales server performs a lookup and learns that Bob is located as bob@number4.sales.company.com. It proxies the the request to the server (user agent server) executing on the host number4.sales.company.com (mes-

[13]SIP URLs are similar to email URLs. An email URL for Bob might be mailto:bob@company.com.

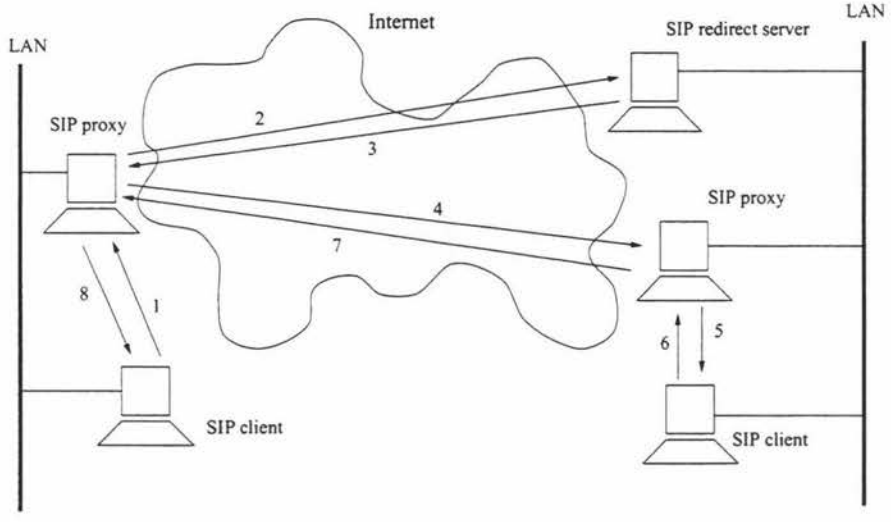

Figure 3.1: An example of call setup in SIP. See text for the message key.

sage 5). Bob accepts the call and the response is returned through the chain of proxies. The return path is noted by messages 6,7 and 8.

H.323 and SIP

The development of SIP was prompted by the criticism of the H.323 protocol for its lack of scalability when signaling on the Internet. Due to the similar uses of SIP and H.323, Schulzrinne and Rosenberg (1998) have compared the two protocols in great detail. Their comparison criticizes many aspects of H.323 including its complexity, lack of extensibility and scalability problems. The criticism of the complexity of H.323 stems from the size of the base specifications of H.323 (736 pages). The binary representation of H.323 is criticized as it is difficult to parse. H.323 uses the abstract syntax notation (ASN.1) for its message encodings. As this is a standardized encoding scheme it is likely that existing parsers can be re-used. The

mere fact that the protocol encodings are in binary rather than textual form does not necessarily mean they are difficult to parse. The C programming language provides flexible bitwise operations that make binary parsing relatively easy. H.323's complexity is also criticized as it is comprised of three sub-protocols.[14]

Furthermore it is claimed by Schulzrinne and Rosenberg (1998) that there is no clean separation between the protocol components of H.323. The example given of a call forward service demonstrates that these protocols are not merged cleanly. H.323 is noted to provide a number of different ways of performing the same task across its two major versions (version 1 and 2). At least three distinct ways to setup a call exist and given that any gateway device will have to support all of them does seem to be unwarranted complexity. The criticism of protocol complexity does seem well justified. H.323 is also claimed to lack extensibility relative to SIP in Schulzrinne and Rosenberg (1998). They note that SIP has learned from the lessons of HTTP (Fielding et al., 1997) and SMTP. Both HTTP and SMTP are protocols that are widely used and have evolved over time. The use of *Require* headers in SIP allows clients to indicate named feature sets that the server must understand. If the server does not understand any of these features it returns an error code and lists the set of features it does understand. This way the client can request limited functionality. Due to SIP's similarity to HTTP, mechanisms being developed for HTTP can also be used in SIP. Among these are the Protocol Extensions Protocol (PEP) which contains pointers to the documentation for various features within the HTTP messages themselves. If this were successfully applied to SIP the protocol would be very

[14]Three protocols are used for simple call processing functions. A number of other protocols are used for more advanced functionality as described in the earlier H.323 section.

extensible indeed. H.323's extensibility mechanisms, on the other hand, are provided through the *nonstandardParam* fields placed in various locations in the ASN.1. These parameters contain a vendor code, followed by an opaque value which has meaning for that particular vendor only. Extensions are therefore limited to the places that a non-standard parameter exists. The vendor cannot provide extra functionality on messages that do not allow a non-standard place holder. For these reasons H.323 does seem to suffer from extensibility problems when compared with SIP. In fact, Schulzrinne and Rosenberg (1998) note that many other aspects of H.323 also suffer from serious extensibility problems.

According to Schulzrinne and Rosenberg (1998) the criticism of scalability of H.323 is apparent in four areas. These are in the contexts of large domains, server processing, conference sizes and quality feedback. If we take the server processing criticism as an example we see that H.323 is certainly limited in scalability. The fact that H.323 uses the transport control protocol (TCP) to carry the data for each of the three protocols it uses means that a server is quickly limited by the number of connections available. Under TCP each socket communicates on a unique *port number* which is essentially an application address. A 16 bit quantity is defined as the port number for TCP connections. This means that there are less than 22 thousand H.323 connections possible per host. A large gateway device would be expected to carry a far greater capacity than this. In contrast, SIP does not suffer from the problems of scalability that H.323 does. In light of these comparisions it seems that SIP is more suited for use as a signaling protocol in Internet Telephony as it is more scalable, extensible and less complex than H.323.

The Media Gateway Control Protocol

One futher criticism of H.323 relates to its requirement that both the call signaling information and media transformation function (converting digital packetized data into analog signals and vice versa) are performed on the same host (Huitema et al., 1999). Under a large number of connections, this will be a limiting factor. MGCP - The Media Gateway Control Protocol (Cuervo et al., 2000) - is proposed to perform the gateway control functionality without this scalability problem inherent in H.323. The reader is referred to Huitema et al. (1999) for a detailed discussion on the interworking of SIP and MGCP.

The central concept behind MGCP is the separation of the media transformation and signaling functions. The signaling functionality is handled by a device termed the *Call Agent.* The call agent uses the MGCP to control the TGW - *trunking gateway* - and RGW - *residential gateway.* These entities and their relationships with the PSTN are shown in Figure 3.2.

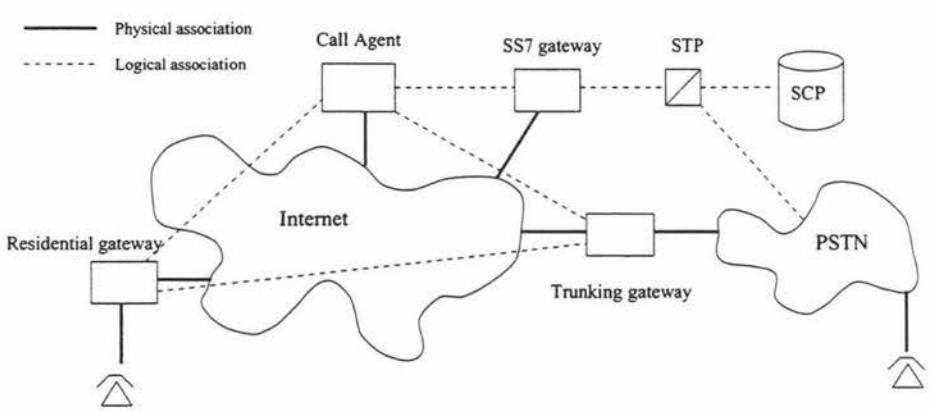

Figure 3.2: The use of MGCP integrating the PSTN and Internet Telephony.

The TGW is responsible for the media transformation function that

converts PSTN voice signals to RTP packets and vice versa. The TGW supports MGCP for two reasons. The first reason is to enable it to inform the call agent of events that it has received from the PSTN. The second reason is to allow the call agent to instruct the TGW to perform some task (for example, connecting trunk 1 to a certain IP address).

The RGW is responsible for capturing events associated with an IP telephony subscriber. Examples of these events include dialing digits, or placing the telephone on-hook. The RGW is also required to support RTP so that it can transmit the voice signal end-to-end. The RGW must, in addition, support MGCP allowing it to send events to and receive instruction from the call agent. An RGW is not limited to this functionality, however. It may support other capabilities depending on the customer's needs. The residential gateway is expected to contain at a minimum one PSTN line, MGCP for setting up calls, RTP for voice communication and a network interface to transmit and receive MGCP and RTP. It is expected that a subscriber's telephone will plug into the RGW and that the RGW will in turn be connected to the phone line.[15]

The call agent controls both the RGWs and TGWs via the MGCP. Both these gateway devices report events to the call agent and the call agent then instructs the devices on how to continue the call. The call agent also handles the SS7 signaling for the trunks that connect the PSTN with the IP network using the ISUP protocol. Call agents also use the TCAP protocol to communicate with SCPs on the carrier's AIN. This enables the Internet Telephony subscriber to access the same services as a PSTN subscriber.

[15]Huitema et al. (1999) do not give a reason why a telephone line exists at all. The phone line is presumably used in the case of a power failure.

3.3.5 Other IP Telephony protocols

There are a number of other protocols that provide portions of the functionality necessary for Internet Telephony. Three protocols relevant to this work are described in this section. These protocols are the RTP – Real Time Transport Protocol – used for transporting real-time data over an IP network, the RTCP – Real Time Control Protocol – which provides feedback on a RTP session, and the GLP – Gateway Location Protocol – which is used to locate an appropriate gateway when placing calls between the Internet and the PSTN.

The Real Time Transport Protocol

The Real Time Transport Protocol is used to stream packetized voice data over an IP network. In a packet based network, the entire process of transporting media involves packetizing the data generated by a media encoder, sending these packets across the network and recovering the bit-stream at the receiver. Packets may be lost during transmission, they may be delayed by varying amounts of time and they can be reordered by the network. In order to accurately re-assemble the media stream, timing information must be conveyed by the transport protocol so that the receiver is aware of the variability in the network delay. RTP does not attempt to solve the problem of variable network delay, packet loss or packet reordering. Instead it allows the receiver to recover from these conditions. This is enabled by RTP's header information. This header contains the following:

- Sequence numbers. These are used to detect packet loss and reordering.

- Time stamps. Each packet contains the time it left the media source. This information can be used by an algorithm at the receiver to cater for variable network delay.

- Payload identification. Speech and video coders and decoders differ in their ability to function correctly under various data loss conditions. Hence it may be useful to change the encoding algorithm used during transmission. Therefore each packet contains a description of the algorithm used to encode its data.

- Frame identification. Video and audio are managed in logical units termed *frames*. In order to indicate to the receiver the beginning of a new frame a frame marker bit is provided.

RTP also supports multicast operation. The Click-to-Dial service which is discussed in the present study requires the use of RTP. The Internet Call Waiting service may also use it.

The Real Time Control Protocol

The Real Time Control Protocol may be used in conjunction with RTP. RTCP provides the following additional information to session participants:

- Quality of service feedback: Receivers may provide the sender with a description of the quality of their session. This includes the number of lost packets, variable delay, and round-trip delays.

- Inter-media synchronization: Audio and video are often carried in separate streams to provide flexibility. These streams need to be synchronized.

- Session control: RTCP allows session participants to indicate they are leaving the session or to send small notes to each other.

The Gateway Location Protocol

If an Internet Telephony user wishes to call a party located on the PSTN, a gateway device must be used. This device must be capable of both (1) converting packetized audio to an analog signal representing the voice data and (2) providing the appropriate signaling necessary to set up a call to the PSTN subscriber. A large number of gateway devices may be used in the future. Each of these gateways may be used to call a portion of the range of valid phone numbers. The GLP – Gateway Location Protocol – is used in returning the appropriate gateway for a particular call. The architecture for GLP assumes that a number of Internet Telephony domains exist, each of which is under the control of an authority. Each of these domains has one or more PSTN/IP gateway devices. Each domain also has one or more location servers. These location servers are aware of the gateways in their own domain by the means of a protocol termed an *intra-domain protocol.* This protocol propagates information from gateways to location servers within a domain. It is assumed that a single domain will not have access to enough gateways to complete calls to all possible telephone numbers. Hence users of one domain may make use of gateways in another domain. This is expected to require contracts between the domain authorities. Once the appropriate contracts exist, the gateway information for a particular domain may be passed to another domain. The GLP is used to exchange the inter-domain gateway information. Thus it enables a location server to gather a database of gateways in other domains. Each entry in this database is comprised of

the IP address of the gateway, a range of numbers the gateway is able to place calls to, and attributes that describe the gateway. Examples of these attributes include cost information, supported encoding algorithms, signaling protocols and port numbers. At the time of writing the GLP is still in the specification stage (Rosenberg & Schulzrinne, 1999; Rosenberg et al., 2000).

3.3.6 Implementation of IP telephony services

The subject of the implementation of Internet Telephony services is the topic of much current work and some recommendations have already been made. A significant portion of the work of Rosenberg et al. (1999) involves decisions about the location of the service implementation, the means by which an implementation interfaces with the protocols that are used to deliver the services, and how much control a service implementation has in the call. They present requirements for programming Internet Telephony services and show that two solutions are required. The first solution enables service creation by trusted users, the second allows service creation by untrusted users. The result of this is the use of a common gateway interface for trusted users, and the Call Processing Language for untrusted users. Rosenberg et al. (1999) also discuss the programming of Internet Telephony services using SIP. They recommend the use of a common gateway interface for SIP services. The common gateway approach is recommended due to the use of a common gateway interface in HTTP. Indeed,

> "SIP's similarity to HTTP makes applying CGI to Internet Telephony straightfoward and advantageous" Rosenberg et al. (1999).

One of the problems, in our opinion, with the approach of Rosenberg et al. (1999) is that they discuss the programming of *SIP services* and subsequently present the SIP CGI as a mechanism for *Internet Telephony* services. This is unlikely to be the best way of addressing this issue as SIP is not the only signaling protocol that may be used by an Internet Telephony service. Furthermore Rosenberg et al. (1999) introduces a new language for call processing for untrusted users. If untrusted users are supported there is a proliferation in services that can be offered. However, whether a new language should be required to produce these services is debatable. The widely used Java programming language may instead be used by untrusted users to program Internet Telephony services through the use of the JAIN security APIs.

Thus far we have discussed issues in the field of Internet Telephony including the integration of the PSTN and the Internet, signaling protocols and other technologies that are used in Internet Telephony that are relevant to JAIN. This includes implementation of Internet Telephony services, the Real-time protocol, the Real Time Control Protocol and the Gateway Location Protocol. Service developers may develop Internet Telephony services in Java using JAIN API's for protocol access. However, JAIN is not the only Java based telephony initiative. The next section discusses other Java based telephony initiatives.

3.4 Other Java-based Telephony initiatives

3.4.1 Java Telephony API

The JTAPI – Java Telephony API (Sun Microsystems, 1999h) provides Java APIs which allow CTI – Computer-Telephony Integration – applications

to be developed using the Java programming language. These applications are able to run on any operating system and processor combination which supports the Java runtime environment and contains a conforming JTAPI provider implementation. The JTAPI provider is a layer of software which provides the JTAPI programming interface over an existing telephony programming interface.

CTI applications and telecommunications services are notably different. Telecommunications services tend to be offered by carriers with the purpose of gaining market share and producing further revenue. They are hosted on machines that are connected to the carrier's signaling network whereas CTI applications provide functionality that is enabled by combining a telephony interface (such as a telephone style card or an ISDN terminal adaptor that is attached to a computer) with an Operating System so that application software can access the telephone network as a user. This allows a host of applications to exist.[16]

The intentions of JTAPI and JAIN are quite different. However, they do overlap slightly in IP telephony because JAIN provides access into IP telephony protocols. This would enable CTI applications to use a JAIN interface for an IP signaling protocol. In this case the CTI application performs signaling. Obviously the boundary between services and CTI applications is blurred in the case of Internet Telephony.

The JTAPI architecture defines a set of core packages which provide for basic telephone use. A number of additional packages are also available for JTAPI. The core package defines an elegant call model that is an Object

[16]For example, a CTI application may monitor the stock exchange placing calls which play messages informing the called party of significant changes in stock levels, or it may monitor the local weather and play the information it has gathered to a calling party.

Oriented abstraction of the various states of a call. This call model may be used by other technologies. For instance, the Parlay Call model appears to be very similar to that of JTAPI. JTAPI is limited in its use of a telephone number as an address for calls. This limitation becomes apparent in combination with Internet Telephony when using signaling protocols such as SIP (SIP uses a URL as a destination address) and H.323 (H.323 uses an IP address as a destination address). JTAPI may need to be extended in the future for use with Internet Telephony.

3.4.2 The "Softswitch"

A number of equipment vendors are introducing products that provide generic call control models with support for ISUP and IP based signaling protocols. Such a product is commonly referred to as a *Softswitch.*[17] Services for Softswitches are commonly developed in Java. Softswitches enable services that may use either or both the Internet and the AIN in their operation. The use of a generic call control model and Java make Softswitch similar to JAIN. However, unlike the JAIN initiative, services written for a Softswitch are not portable as each Softswitch provides its own API for call control.

[17]Examples of Softswitches include http://www.softswitch.org, Lucent's Softswitch Platform at http://www.lucent.com/software/CNS/softswitch.html or Xybridge's SmartSuite at http://www.xybridge.com/products.html.

Chapter 4

Service examples using JAIN

4.1 Introduction

In order to illustrate the value of the JAIN proposal, two services are described that are capable of executing in the same fashion on multiple vendor platforms without modification to the service logic. Both of these services operate using both the PSTN and the Internet showing the suitability of JAIN for use in heterogenous networks. These services are Internet Call Waiting and Click-to-dial. Each of these is discussed in turn. An Internet Call Waiting prototype, in fact, has been developed conforming to the architecture presented in this chapter. A summary of the implementation of this service is presented in Appendix A.

4.2 Internet Call Waiting

Internet Call Waiting (ICW) is a facility that allows subscribers to receive notifications of incoming telephone calls while they are connected to the Internet. It is best illustrated by the following scenario:

> Imagine that Alice is connected to the Internet via her modem. During this time, Bob tries to phone Alice. If Alice didn't subscribe to any additional services, she would simply be oblivious to the incoming call. Bob would have just received a busy signal and would have had to hang up. If Alice had subscribed to Call Waiting, then she would have received an indication on her modem about the incoming call. At this point she has the option of hanging up her Internet connection and taking the incoming call. If she had subscribed to Internet Call Waiting, then Alice would probably see a window pop up on her PC informing her of the incoming call with information about who the call originated from. She might also be presented with options which allow her to either take the call through her computer's multimedia hardware, or redirect it to a voice-mail box.

ICW, as described above, is already available to many telephone users as several carriers have implemented it (Rizzetto & Catania, 1999).[1] However, these implementations are not in the public domain. In Nortel's implementation of ICW, carriers and Internet Service Providers (ISPs) are required to install an ICW server and proprietary software from Nortel on their networks. The ICW server will be connected to both the AIN (via an ISDN line) and to the Internet. When a user who is connected to the Internet receives an incoming call, it will be routed to the ICW server instead. The server will obtain the relevant details regarding this call over the SS7 network and will proceed to notify the called party via the Internet using its

[1] See, for example, Lucent's press release at http://www.lucent.com/press/0598/980512.nsa.html or Nortel's online advertisement at http://www.nortelnetworks.com/products/01/icw/index.html

Internet connection. Figure 4.1 shows an example of a dialog box that is presented to the called party by a commercially available ICW service – Nortel's.

Figure 4.1: Internet Call Waiting GUI

Using this implementation of Internet call waiting, the ISP is required to purchase Nortel's ICW server. As the implementation is proprietary, the ISP is not able to differentiate itself by adding value to the service. Furthermore, if the ISP decided that the ICW server was not scalable enough, and that the service should run on another platform, it cannot use this service on the new platform.

If the source code was available, the ISP might attempt to port the service to the new platform. However it is highly likely that porting will be difficult due to platform dependent programming interfaces and the need to port vendor specific libraries that have been used in the implementation. For example if the database used to store the telephone number to IP address mappings is not available on the new platform then significant work might be required to accommodate a new database. We present below an architecture for ICW using JAIN. This architecture is more customisable by an ISP and may execute without modification on many vendor's platforms as it accesses

the SS7 stack through a JAIN API.

4.2.1 ICW implementation using JAIN

One obvious architecture for the ICW service as described above is to have the carrier maintain a table mapping telephone numbers that are currently connected to the Internet, to the IP addresses that they have been assigned, and for the carrier to contact the called party themselves. The ICW subscriber, in the example, Alice, is required to install and run software on her PC that would listen for messages from the ICW server maintained by the carrier. The port number (application address) where Alice's system expects this message could be pre-arranged and thus well-known. But how does the ICW server know what IP number Alice has been dynamically assigned by her ISP? The solution is simple if Alice's carrier also happened to be her ISP. But there is no reason to suppose that this is always so. Alice might choose an ISP that is different from her carrier and still continue to subscribe to her ICW service. One way of addressing this problem is to populate the carrier's telephone-number to IP-address map by requiring the call recipient's (Alice's) application to contact the carrier upon connection to her Internet Service Provider (ISP). Alternatively, the ISP could inform the carrier that a particular IP address has been assigned to a particular phone number. But both the above approaches require cooperation between multiple parties. Furthermore, if the ICW server were installed by the carrier, it would have to manage a potentially large mapping between phone numbers and IP addresses since the size of this map is proportional to the subscriber base of the carrier.

Alternatively, if it is the ISP who hosts the ICW server, then the carrier

must notify this ICW server at the ISPs premises of incoming calls to Alice, via the SS7 network. The ISP will then proceed to notify Alice of this call since it knows the IP address it had assigned to Alice. However, in this case, the ISP is required to purchase and install specialized hardware and software for interacting with both the AIN and the Internet. Further, both the user interface and the flexibility of the ICW service are limited by facilities already built into the ICW server purchased by the ISP. There is little scope for an ISP to differentiate itself from other ISPs in terms of the form and function of the ICW service. As we can see, a cleaner and more elegant architecture is possible as described below.

The ideal solution would be for ISPs to purchase a basic ICW service from the carrier and then offer enhanced versions of this service to their customers. This provides scope not only for ISPs to differentiate themselves from each other based on their qualities of service, but also the ability to purchase the ICW server from any third party developer. Since ISPs themselves are typically permanently connected to the Internet with static IP addresses, they may register this IP address with the carrier upon their purchase of the ICW service. Returning now to our previous example, suppose that Alice's preferred ISP is different from her phone carrier and that she has obtained the ICW service through this ISP. When Bob calls Alice during a time that she is dialed into this ISP, the carrier is able to detect not only that Alice's phone is busy, but also that she is also dialed into an ISP that has subscribed to the carrier's ICW service. At this point the carrier proceeds to notify the ISP of this incoming call, but does this over the Internet rather than over SS7 since the ISP has registered its static IP number with the carrier.

Furthermore, this information from the carrier to the ISP can take the

form of a single packet that is sent to a well known port on which the ISP's ICW server is listening. This server itself, being only software, the ISP has the freedom of either designing itself, or purchasing from any third party software developer. One may wonder at this stage how this circumvents the problem of large maps at the carrier's end. The ISP typically uses one physical incoming telephone line for each Internet connection it provides. Thus, if the carrier is to notify the ISP over the Internet rather than the telephone user, doesn't it still need to maintain a table that maps each of these phone numbers to the ISP's registered IP address? For example, assume that an ISP has n available modems for dialing into. Each modem necessarily connects to a line with a unique telephone number. Call these phone numbers $P_1, P_2, \cdots, P_n$. Assume also that Alice has dialed into P_1, Andrea into P_2 and so on. It seems then that the carrier now has to maintain a table mapping each of the P_i into the ISP's static IP address. Instead of the map between the numbers of the called parties and their IP addresses, the carrier now maintains a map between the numbers that the called parties are dialed into and the IP addresses of the ISPs. It seems that there has been no improvement in the size of the mapping. However, this is not so in practice since ISPs tend not to advertise physical phone numbers for their customers to dial into, but rather 800 numbers. Usually it is the SCP on the carrier's network that resolves these 800 numbers to physical phone lines depending on various factors. If this is the case, the carrier does not have to maintain a table mapping each physical destination phone number into the ISP's IP address, but rather a table mapping the 800 numbers of each ISP into their IP addresses regardless of the physical line that each 800 number eventually gets resolved into. The SCP can easily determine if a called party is dialed into an 800 number that is registered for the ICW service

and proceed to notify the corresponding ISP accordingly. This means that the size of the mapping from phone numbers to IP addresses at the carrier is now proportional to the number of ISPs in its subscriber base and not to the size of the subscriber base itself.

There are thus three main parts to the proposed ICW service. The first part resides with the carrier. We call this the carrier part. This is responsible for the following:

1. To resolve 800 numbers of ISP's into their static IP addresses

2. To notify the ICW-subscribing ISP's via the Internet when a user who is dialed into their 800 number receives an incoming call.

The second part resides with the ISP's. We call this the ISP part. This is responsible for:

1. Receiving information from the carrier via the Internet regarding incoming calls to their dialed-in customers and

2. Formulating and dispatching an appropriate message packet to a client application listening on the customer's PC.

The third part is the ICW user interface which resides at the ISP's customer's premises. This could be, for example, an application that listens on a well known port for alert messages from the ISP. The ICW notification may be one of such alert messages.

Figure 4.2 shows a pictorial representation of these different parts in the proposed Internet Call Waiting service. In the following sections we elaborate further on the various components of the ICW service.

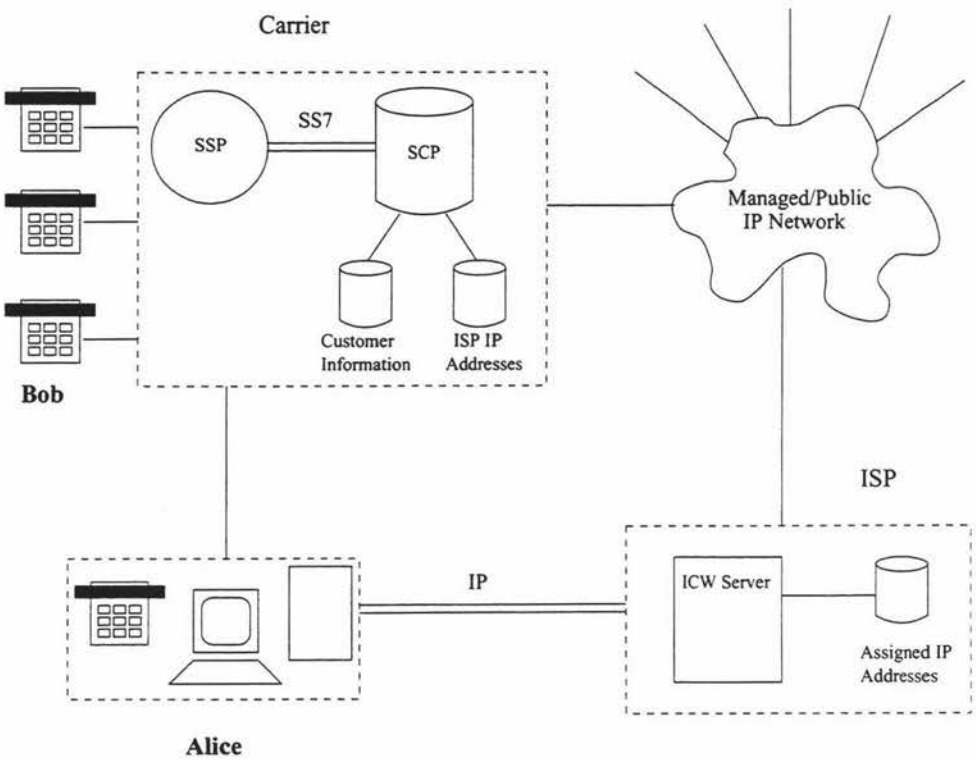

Figure 4.2: ICW architecture using JAIN

4.2.2 The Carrier Part

The service logic for ICW primarily resides on the SCP within the carrier's network. The SSP provides the SCP with call information which consists of the number being called and the calling number. Normally, the SCP determines the appropriate course of action upon receipt of this message from the SSP. For instance, if the called number is not connected to an ISP that is registered for the ICW service from the caller, then the SCP may execute the normal service logic for the call.

If the SSP determines that a certain number being called is busy, its

normal response at this stage is to notify the SCP of this situation. This is required because the SSP is typically unaware of what services the called customer may have subscribed to. It may be the case that the customer has not subscribed to any additional services and that the SSP simply plays a busy tone back to the caller. But before this happens, it must ensure that this is indeed the case by checking with the SCP. Upon receipt of this notification from the SSP, the SCP first determines the appropriate service to execute for the incoming call. Again, note that the selected service could just be one that plays the busy tone back to the calling party. Assume, however, that the service happens to be ICW, which the customer is subscribed to indirectly via the ISP.[2] This implies that the SCP had successfully determined that the customer was dialed into an 800 number which could be located in its database of ISPs registered for ICW notification.

At this point, the SCP will construct a notification packet to send to the customer's ISP over the Internet. It is able to do this because the ISP's static IP number is available to it through the ISP database. This packet can be as simple as the carrier pleases. It need contain no more than the called and calling numbers together with a checksum and other authentication fields, or it might be an elaborate frame containing in addition the calling party's identity and information as to whether the called party has also subscribed to other relevant services such as voice-mail at the carrier's end. Returning to the example with Alice and Bob, we may roughly assume the following sequence of events to happen:

[2]We assume that customers subscribe to ICW notification from their ISPs, who in turn register these customers with the carrier. Subscription to the service may involve signing an agreement that permit the ISP to obtain calling party numbers or identities from the carrier.

1. The SSP determines that the dialed phone number is busy. It then constructs a TCAP query, containing both the dialed phone number and the number which the dialed phone number is connected to. This is the number that Alice dialed when she connected to her ISP. It is important to note that this is the non-translated 800 number.

2. The SCP receives a TCAP query from the SSP. It uses the information contained in this query to decide whether to start the ICW service. The ICW service is started if the number Alice dialed corresponds to an ISP that is registered to deliver the ICW service and Alice had subscribed to the ICW service through her ISP. This is achieved by placing a request to the ISP IP addresses database, which maintains a set of key-value pairs, each containing an 800 number and an IP address. The 800 number is the access number for an ISP, and the IP address is the address of its ICW server. If there is no entry for a particular number it means that the dialed party (Alice) is not currently connected to an ISP registered to deliver the ICW service.

3. Optionally, the SCP may also attempt to resolve a name for the calling phone number if the ISP had registered for extended caller information. In this case the name 'Bob' would be returned. However if a name does not exist for the dialing number, an appropriate string is returned in its place. This might occur if the dialing party is subscribed to a different carrier, or is calling from another country.

4. A message is sent over the Internet to the IP address gained in step (1). This message contains the name of the caller, the caller's phone number and the destination phone number.

Method	Method call
1	processComponentIndEvent(ComponentIndEvent event)
2	updateMessage()
3	processDialogueIndEvent(DialogueIndEvent event)
4	parseMessage()
5	getIpAddressFromPhoneNumber(PhoneNumber IspAccess-Number)
6	getNameFromPhoneNumber(PhoneNumber dialer)
7	constructCallNotification()
8	sendMessage(CallNotification notification, InetAddress destination
9	responseReceived(byte[] IspResponse)

Table 4.1: Possible method calls for a carrier part implementation. See figure 4.3 for the corresponding sequence diagram.

5. The carrier waits a predefined period of time for a response. If this interval expires and a response has not been received by the carrier it will fall back to normal call processing. If a response is received during this time period, then it will contain instructions on the appropriate action to perform. These actions could include redirecting the caller to voice mail, asking the caller to disconnect and wait for Alice to respond, or connecting Bob to Alice using VoIP and a PSTN/IP gateway device.

6. After Bob disconnects, the carrier's responsibility in the handling of this instance of the ICW service can be considered finished.

This process is shown as a UML sequence diagram in Figure 4.3. Each number in the figure represents a method call. These are shown in Table 4.1.

The execution of the service is started when the SSP sends TCAP data to

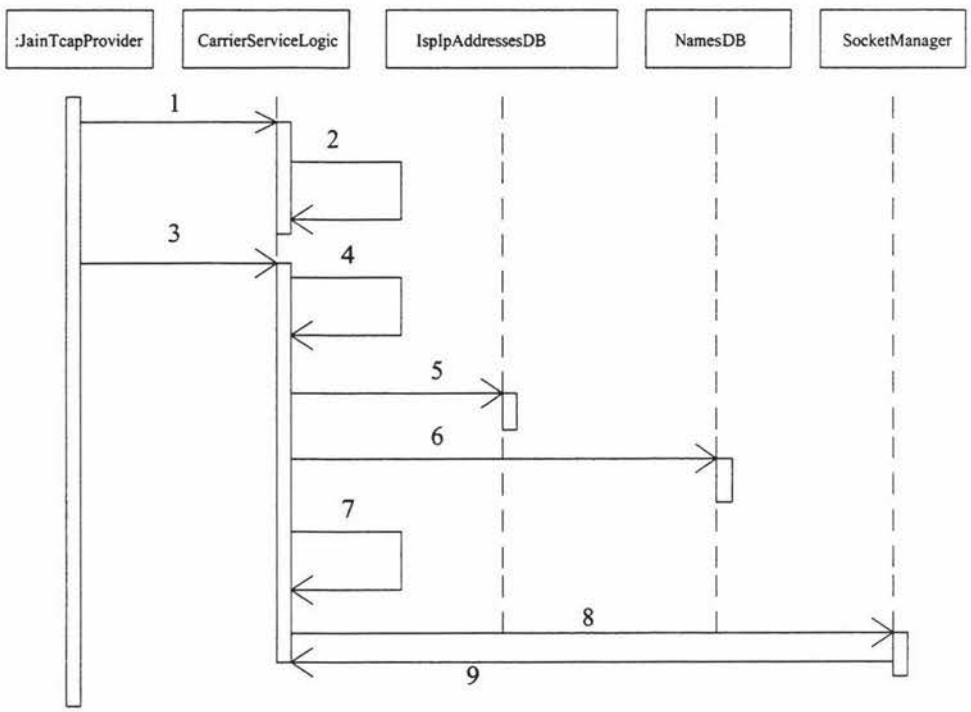

Figure 4.3: A UML sequence diagram of the Carrier Part service logic that is intended to be illustrative of a possible implementation. Each number represents a message being passed between objects. Refer to the text for an explanation of these. It is important to note that the actual event ordering generated by the *JainTcapProvider* can differ upon receiving different TCAP messages.

the SCP on a busy trigger. This message is received by the SS7 stack on the SCP. The SS7 stack vendor implements several objects and interfaces present within the Jain TCAP API (Sun Microsystems, 1999c). The service receives events which encapsulate SS7 messages from the *JainTcapProvider*, which ties to the SS7 stack. Once the service logic has received the *DialogueIndEvent* it places requests to the database wrappers for the ISP IP addresses database and the caller name database represented as *IspIpAddressDB* and

NamesDB in the sequence diagram. These look up the required information to assemble a call notification message. This message is then sent to the appropriate ISP.

4.2.3 The Internet Service Provider Part

The Internet Service Provider is responsible for appropriate handling of call notification messages that it receives from the carrier. The first step in the processing of such messages involves parsing them to extract the key fields. At a minimum the ISP will extract the called number (the phone number of the customer who is currently dialed into the ISP's modem banks) and the phone number of the party calling this number. It may also optionally contain extended information if the ISP had registered for it. For instance, it may be the case that the ISP is also notified that the called party has a voice mail box at the carrier's end, in which case, the ISP is able to offer to the called party the choice of redirecting the caller to this voice mail box.

Once the ISP has parsed the the incoming notification packet, it determines the dynamic IP address that it had allocated to the customer who is being called. This, it obtains by looking up the called phone number in its database of phone number–IP address key-value pairs. This set of mappings is represented diagrammatically in Figure 4.2 as the *AssignedIpAddresses* database located within the ISP's Internet Call Waiting server. A new mapping is inserted by the ISP when a customer connects and is assigned an IP address and it is removed when the customer disconnects from the ISP.

After it has determined the dynamic IP address it then proceeds to construct a notification packet similar to the one constructed by the carrier

and proceeds to send this over IP to the customer. Again, this step may be as simple or as elaborate as the ISP pleases. For instance, the ISP may simply decide to re-route the carrier's message by changing the destination IP address to that just resolved for the customer and placing it back on the output queue, or it may decide to construct an entirely new packet consisting of several additional fields that bear relevance to the call recipient part that resides on the customer's computer.

At this stage the ISP awaits information from the call recipient part as to its next course of action. This will depend on the user's response to dialog boxes or it may be selected automatically from user settings and preferences by the call recipient part. For example, the called party could have a local database instructing the call recipient part to perform specific actions for particular incoming caller identities.

If such a message is received from the call recipient part within a reasonable timeout period, the ISP will perform the action requested. It may be that the called party requests to be disconnected from the ISP and connected to the caller or it may be that the called party wishes the caller to be redirected to a voice mail box. Alternately, the called party may wish to receive the incoming call using VoIP. After it has suitably responded to this message from the called party, the ISP may consider its responsibility in handling this instance of the ICW service completed.

Returning again to the earlier example, the following sequence of events may roughly be assumed to take place after the ISP received a call notification for its customer from the carrier.

1. The ISP reads the message and separates its contents. The content of this message is discussed in section 3.3 and contains the calling parties

Method	Method call
1	processCallNotification(byte[] callNotification)
2	parseCallNotification(byte[] callNotification)
3	getIpAddressFromPhoneNumber(PhoneNumber dialed-Number)
4	constuctPopupMessage()
5	sendMessage(PopupMessage notification, InetAddress destination)
6	responseReceived(byte[] userResponse)
7	notifyCarrier(byte[] desiredAction)

Table 4.2: Possible method calls for an ISP part implementation. Refer to Figure 4.4 for the corresponding sequence diagram.

phone number and the destination phone number at a minimum.

2. It uses the destination phone number from the data gathered in step (1) to request the dynamic IP address of the call recipient using its assigned IP addresses database.

3. It constructs a call notification packet. Again, this could be very simple containing, at a minimum, the calling parties phone number and optionally the caller name together with a checksum and fields for authentication, or it may be more elaborate containing information pertinent to other services that the customer may have subscribed to.

4. It sends the newly constructed call notification message to the destination IP address gained in step (2).

This sequence of events is represented by the sequence diagram in Figure 4.4. Each of the events in the figure are method calls. These are described in Table 4.2.

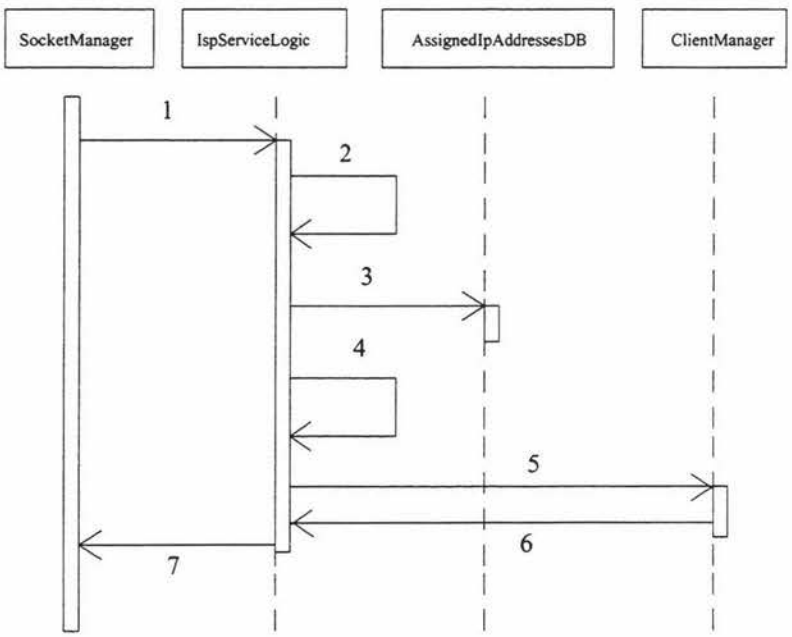

Figure 4.4: The sequencing of service logic within a possible Internet Service Provider part. Each of the numbers in this diagram represents a message being passed between objects. These messages are explained in the text.

At this point in time the ISP has delivered the call notification to the end user. It must now wait a predefined period of time to receive the end user's response. If the end user has not responded before this time period expires it informs the carrier of this event. The carrier is then responsible for informing Bob that the destination party is not available at the current point in time. If a response has arrived from the end user within the predefined period of time it will contain the end user's selection. The carrier is informed of the appropriate action by the ISP.

4.2.4 The Call Recipient Part

A portion of the logic of the ICW service resides on the call recipient's computer. This portion of the service is responsible for the appropriate handling of the call notification messages it receives from the ISP as described in section 3.4 and sending a timely response back to the ISP. The call recipient part performs a series of actions upon receiving an incoming call notification message. These are as follows:

1. Parse the incoming call notification.

2. Notify the user that a someone is attempting to call them. During this notification the user should be presented with options which allow them to select the desired action. For example they might see a panel with the options "Foward to Voice-mail", "Redirect to Cell", "Take call". Such a user notification is displayed in Appendix A.

3. Wait a predefined period of time for the user's response. If the user does not respond within this time a default message should be sent to the ISP.

This sequence of events is depicted in Figure 4.5. The numbers on the messages are shown in Table 4.3.

In order to receive these call notification messages, the end user must have a process executing on their PC which listens on a dedicated port number. This software could be implemented in any language that has programming interfaces allowing access to user and IP network interaction. The Java programming language provides both of these and has the additional benefit

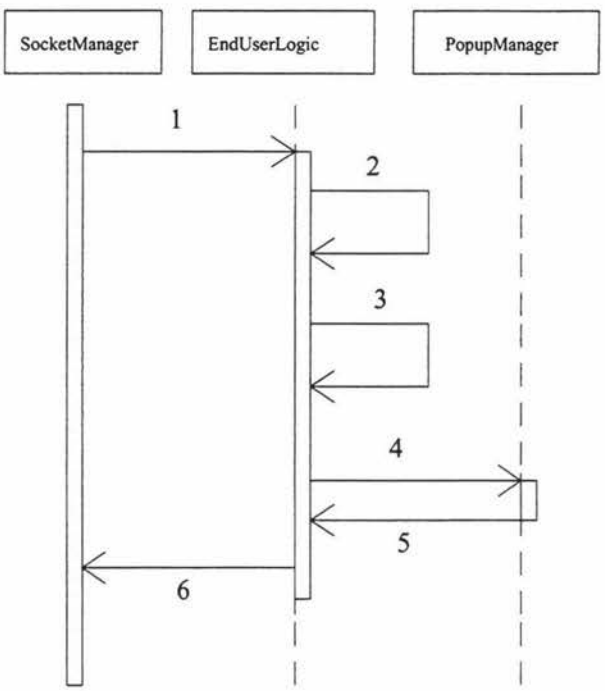

Figure 4.5: A sequence diagram of a minimal end user part application that is intended to be illustrative of a possible implementation. Each number represents a message being passed between objects. These messages are described in the text.

of being cross platform. A further benefit of Java is that of the applet interface which enables Java software to be downloaded and executed by the end user's web browser. Hence this software may be kept up-to-date without intervention by the end user.

Anjum et al. (1999) present a Java framework that allows a user to subscribe to new services and receive the necessary software to handle these services simultaneously. This framework may be used to handle new messages which the ISP sends to a subscriber.

Method	Method call
1	callArrived(byte[] callNotification)
2	parseCallNotification(byte[] callNotification)
3	constructMessage(String dialerName, String dialerNumber)
4	openWindow(PopupMessage incoming)
5	responseReceived(String buttonName)
6	sendResponse(byte[] userResponse)

Table 4.3: Possible method calls for a call recipient part implementation. Refer to Figure 4.5 for the corresponding sequence diagram.

4.2.5 ICW security considerations

In a PSTN telephone call, only the originating and receiving parties and the carriers involved in setting up the call are aware of the details of the call. In the proposed ICW architecture there is an additional party the ISP. It is possible for end users to perceive this as a disadvantage, as their privacy may be imagined to not be held as tightly. However, it is worth remarking in this context that an ISP is usually able to view all of their customers Internet traffic which can carry far more revealing information than a call notification message. To this end, many ISPs have contracts with their customers stating that they will ensure their customers information is held in the strictest confidence. When the call notification is sent over the Internet, it is possible that this information could be duplicated in transit. Hence this duplicated message could be made publicly available. This would allow the call notification message to be viewed by parties other than the destined party. It is also possible to fake call notification messages to various ISP's offering the ICW service. Indeed, such a service would be of questionable integrity. Fortunately, these two problems can be easily solved. The first problem can be overcome by encrypting the information sent, so that even if the information is intercepted in transit, it is not in a readable form without

significant computation taking place on the message. A Java framework currently exists for performing such encryption (Sun Microsystems, 1999g).

The second problem is that of authentication, where the receiving party must be sure that the sending party is who they claim to be. The Java 2 platform contains a mechanism for the digital signing of applets so that the user can be sure that the software that they run has not been tampered with in transit. However, a standard Java API for the authentication of incoming connections does not yet exist at the time of writing. Therefore the author of the software would be required to build a custom implementation of an appropriate authentication technique for use in the service. The Java 2 certificate package within the security API provides a means to implement such authentication. Once these provisions have been implemented the service is no less secure than a normal telephone call.

The presented ICW service demonstrates that a modern telecommunications service may be developed in a portable fashion. Furthermore this service is more flexible than current ICW implementations as it allows the ISP to customize the service hence maintaining brand differentiation. An ICW service has been implemented by several vendors. In the next section we present an architecture for an Internet Telephony service that has not been widely implemented, namely Click-to-Dial.

4.3 Click-to-Dial

4.3.1 Introduction

The CTD – click-to-dial – service allows a subscriber to dial a phone number simply by clicking on it within their web browser. This service is illustrated by the following scenario:

> Imagine that Alice is connected to the Internet via an access device. During this time she wishes to call Bob. Alice searches an online telephone directory for Bob's phone number. When the telephone directory has returned Bob's number she selects it by clicking it to converse with Bob using her PC's multimedia facilities and Internet connection.

In this scenario Alice's web browser is integrated in some fashion with software on her PC that is capable of receiving Bob's speech and relaying Alice's speech to Bob who is on the PSTN.

4.3.2 Extending the Web Browser

The integration of the web browser and the telephone in a current day network can be achieved in several different ways. An obvious choice would be for Alice's web browser to use an application on her machine to call the desired phone number. This approach is not desirable as it is likely to contain both platform and application specific code to integrate the browser and external software. Another technique might involve the downloading of a Java applet from the remote site. The applet would then place the call. This approach is desirable as the service provider can control the software used to place calls. The applet would be required to use APIs for playing and receiving audio. The JMF – Java Media Framework (Sun Microsystems, 1999e) – provides these interfaces. But it is an extension to the Java platform. Web browsers that support the Java platform do not necessarily support the JMF. The effect of writing the applet using JMF is

that it may not run on every platform supported by the web browser.[3]

In light of these problems it is clear that a better approach is needed that does not extend the functionality of the web browser as drastically as those previously mentioned.

4.3.3 A Click-to-dial architecture

Adding functionality to the web browser either by directly integrating it with external software or by using Java applets is therefore limited by portability. The click-to-dial service can be designed in such a fashion that it is not constrained in this way.

This may be achieved by placing the call setup in the hands of a third party service provider. The third party can be notified of the call request through the use of CGI – the Common Gateway Interface.[4] If the URL used to dial the number corresponds to a CGI process on the web server, then the data passed to that process may be used for call setup.[5] This data includes the IP address of the host connecting to the web server and site specific data (such as the parameters passed to the CGI process). The IP address may be used by the CGI process as one of the call end points. It is assumed that the site specific data contains at a minimum the dialed phone number. The call may then be setup between the service provider and the PSTN phone number. Once setup the call can be redirected to the IP address of the PC.

This mechanism is advantageous as the web browser does not require any

[3] www.dialpad.com uses a Java applet to enable Internet users to dial a PSTN number. It is limited to the Win32 platform.

[4] CGI allows data to be passed from a web server to an application program located on the web server. The application that receives the data is termed the *CGI process.*

[5] For example www.carrier.com/dial?phonenumber=1234. The CGI process (dial) receives a parameter (phonenumber) with the value 1234.

modification. However, it will not work when a web proxy is used. This is because the web server is only aware of the IP address of the *connected* host. When a proxy is used the CGI process will not receive the calling party's IP address from the web server. Instead it will receive the web proxy's IP address. A solution to this problem is to require the CGI process to send the web browser a Java applet. This Java applet is passed over the HTTP connection. If this applet directly contacted the CGI process then the CGI process would gain the IP address of the client. This applet may be very lightweight as it is not required to provide any user interface or feedback to the user.

As the web browser does not provide the audio functions necessary to maintain a VoIP call this functionality must be present in an application executing on the PC. A number of applications exist that provide this functionality for a variety of platforms.[6] These are referred to as *telephony software* for the remainder of this discussion. Figure 4.6 provides a high level illustration of the various elements used in this service. It can be seen that the call control process is central to the service. It is responsible for communicating with the web browser, applet, location server, PSTN gateway and the calling party's telephony software. This is discussed in greater detail in the next section.

4.3.4 CTD operation

The CGI process is hosted by a service provider. It is responsible for setting up a call from itself to the dialed party and redirecting the call from itself to the calling party's IP address. This setup process may be shown as a series

[6]For example, see Gnome-o-Phone at http://gphone.sourceforge.net, Speak Freely at http://www.speakfreely.org, or Microsoft's NetMeeting.

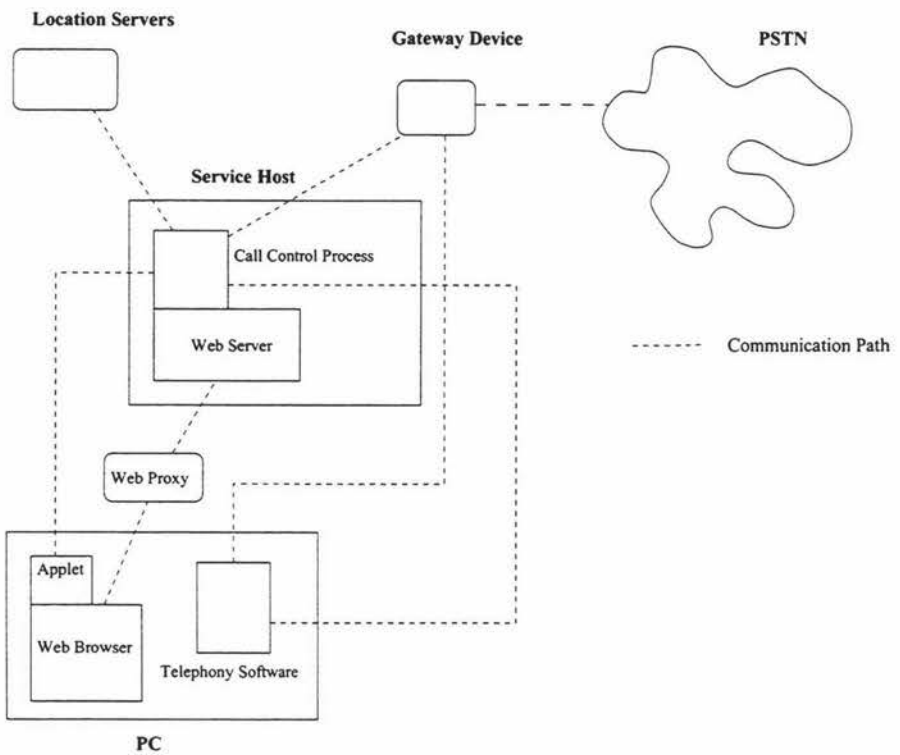

Figure 4.6: The association of various elements used in the CTD service.

of steps:

1. The Web server receives an HTTP request containing a URL. This URL is parsed and it corresponds to a CGI process on the web server.

2. The web server executes the service logic as a child process. This child process may be designed in any manner deemed appropriate. For example it may exist for the duration of the call or it may pass the data it has received to another process that is responsible for call management.[7] We will term the process that handles the call the *call*

[7]The call control process may scale highly if distributed across a number of hosts.

control process.

3. The call control process sends a Java applet to the dialing party's web browser using CGI.[8]

4. The call control process waits for a connection from the dialing party and receives the IP address of the dialing party from this connection.

5. A request to the Gateway Location protocol is made. This request returns an appropriate gateway device. This step may run in parallel with the two previous steps.

6. A call to the dialed number is setup through an appropriate gateway device. This call is then redirected by the call control process to the dialing party's IP address. The dialing party may be informed of progress through the use of CGI.

Once the call has been redirected, the service provider is no longer responsible for the call. Disconnection is negotiated between the gateway device and the calling party's telephony software through the use of a supported signaling protocol. This procedure is complicated by the existence of multiple gateway devices and multiple signaling protocols supported by both the gateway devices and the various telephony software packages that exist.

The appropriate gateway device for any particular call depends on the signaling protocol and codecs supported by the calling party's telephony

[8] A CGI process' standard output is often redirected to the client web browser's data socket. Hence dynamically created data may be passed to the client web browser.

software.[9,10] This information must be known by the call control process so that it may pass it as a parameter to the gateway location protocol when requesting an appropriate gateway device. This information may be collected prior to a call when the subscriber registers for the service. For example, the service provider may require that service subscribers enter a user-name and password when using the service. The user-name and password may be used as an index to retrieve the appropriate information about the customer's supported signaling protocol and codecs.

4.3.5 CTD and JAIN

The Click-to-dial service may be implemented in a number of ways. There are definite advantages to implementing it within the JAIN framework. These advantages are portability, the ease of integration with CGI (using either Java servlets or Java Server Pages), and a host of standard APIs that exist for using a number of technologies such as database and network access, and distributed object technologies such as CORBA or RMI. As can be seen in the above framework the element most likely to use a subset of the JAIN APIs is the call control process.[11] This call control process is required to perform two high level signaling operations. These are call setup and call redirection. It should also be noted that these signaling operations may be required to operate using three different signaling protocols. The logic of the call control process would likely be much simpler if the JAIN Call Control

[9]This signaling protocol may be H.323, SIP or MGCP.

[10]A codec performs the processes of encoding and decoding. In this case a digital packet stream will be decoded and a voice stream will be encoded.

[11]The telephony software might be implemented using a JAIN protocol API however there are several implementations of telephony software that currently exist. Hence this is not explored.

model were used for signaling. Furthermore if the call control model were used then the service would require very little modification if any, in the case of modification of underlying protocols, or if a new signaling protocol was introduced that required support. As mentioned earlier, the call control process places a request to the Gateway Location Protocol. The use of the gateway location protocol would require one or more Java classes to access GLP.

4.3.6 CTD security considerations

The CTD service operates between the PSTN and the Internet. There are a number of issues that must be addressed in order to ensure that the integrity of the service is upheld.

The call control process is responsible for call setup. This process may be run by a third party service provider. This is a similar situation to that of the Internet Call Waiting service, that is an extra party (the CTD service provider) is aware of the destination phone numbers that a dialer may call. This may not be an issue if a contract is signed between the service provider and the service subscriber ensuring that the subscriber's information is held as confidentially as possible.

In this service, the dialed number is passed over the public Internet to the web server. The Hyper-Text Transfer Protocol (HTTP) used between web servers and web browsers passes data in plain text form. Hence the dialed number may be easily gained by an external party. A solution is to use the HTTP over Secure Sockets Layer (SSL) approach that is in widespread use.[12] In this fashion data passed between the web browser and the web

[12]HTTP over SSL is termed HTTPS

server is encrypted.

The call control process requires knowledge of the calling party's IP address. This is gained by the use of a connection to the call control process from a Java applet executing inside the web browser. It is possible that another party could connect to the call control process *before* the calling party's Java applet does and receive calls destined for the calling party. Hence the call control process is required to authenticate the incoming connection. This may be achieved using a key based challenge where the Java Applet is passed along with a key over CGI by the call control process. It then uses this key to authenticate itself.[13,14] As HTTPS is used between the web server and web browser this Applet and key are not readable without significant computational effort.

The voice data is transmitted in some encoded fashion using RTP – the Real Time Protocol. This encoding is not commonly a cryptographic encoding, rather a compressed encoding. Hence it is possible that the voice stream may be duplicated by external parties. The encryption of the voice stream may help this problem however it is vital that this encryption and corresponding decryption does not alter the real-time characteristics of the data. The solution to this problem is presently outside of the scope of this work.

[13] A number of key based challenges exist. Schneier (1995) discusses numerous cryptographic security techniques.

[14] The web server may maintain the various private and public keys per subscriber in the same manner as it stores information on signaling protocol and codec used by the subscriber.

Chapter 5

Summary and Future Work

5.1 Summary

The telecommunications industry is experiencing rapid expansion and organizational change. In this competitive environment participants who can continue to offer new services and efficiently deploy those services will fare best. To ensure that their solutions are chosen ahead of others, service developers should be regarding the portability of the new services they offer as critical. The resulting products may then be offered to all vendors. The best interest of carriers will be served by deploying architectures that provide the greatest potential for service developers to build upon. If they do not, they risk being shut out of the provision of new services, or having to offer them at a competitive disadvantage. The JAIN approach is an attempt to meet these needs by introducing open, standard programming interfaces for portable service development. Our work focuses on the JAIN approach as a solution to the problem of the lack of service portability. Areas of research that are related to JAIN within the field of telecommunications services and

network architectures have been identified and discussed. These areas included the problem of feature or service interaction, the Telecommunications Information Networking Architecture (TINA), Internet Telephony, and the integration of Internet Telephony and the AIN. From this study, several recommendations pertaining to JAIN have been made. These recommendations are areas of future research we have identified. Other Java-based initiatives for telephony were also discussed. These included the Java Telephony API and the emergence of the Softswitch among equipment vendors.

To illustrate the suitability of the JAIN framework, the architecture of two services – Internet Call Waiting and Click-to-Dial – were proposed. An Internet Call Waiting prototype was implemented in approximately 3,500 lines of Java source code.[1]

5.2 Future Work

The use of compatibility test suites for the conformance testing of a JAIN API is arbitrarily rigorous. A particular API test suite may be less thorough than another. This may be perceived as a disadvantage as the quality of conformance of API implementations may be compromised due to non-standard testing. The application of a standardized testing technique to JAIN may improve the quality of API implementations. It may be beneficial if the TTCN approach is used to test Java interfaces. We believe the work of Schieferdecker et al. (1998) on the application of TTCN for the testing of CORBA components to be a suitable starting point.

Implementations of the JAIN protocol APIs are likely to use existing protocol stacks, rather than redeveloping a protocol stack in Java. The

[1] An overview of the prototype is given in Appendix A.

relative performance of a native application using the stacks API in comparison with a Java process using the stack through a JAIN API should be determined. Once this is determined an investigation of the performance of different techniques to communicate from a Java process to the stack should be performed. These techniques may involve the use of the Java Native Interface, CORBA, or a custom protocol between the Java and C/C++ program.

Feature Interaction is considered a significant problem for the introduction of new services to a carrier's network. The integration of various suitable techniques for combating feature interaction should be applied to the SLEE - Service Logic Execution Environment. This environment is likely to include a large portion of third party software components that the service developer does not have direct control over. Hence the minimum that the SLEE should provide is an application of techniques that detect service interaction. In this fashion the service developer may be able to modify the architecture of the service to prevent this problem. A survey of feature interaction techniques is available (Keck & Kuehn, 1998). We believe it to be a good starting point for work in this area.

Telecommunications services may have restrictions placed on the execution time of particular tasks. Approaches used in hard and soft real-time software fields should be analyzed and potentially applied to the SLEE in a platform independent fashion. This approach may require that the Java Virtual Machine displays appropriate real-time behavior. If this were the case then a number of mechanisms provided by a JVM that make program execution non-deterministic will need to be researched. These include:

- Just In Time (JIT) compilation where portions of Java source code are

interpreted by the Java Virtual Machine (JVM) and other portions are compiled into processor native instructions. This means that portions of code may execute faster than other portions of source code. In depth knowledge of the impacts of JIT and other code optimizations made by the JVM may help in the development of a Service Logic Execution Environment that provides predictable service execution.

- Garbage collection that is inherent in the Java programming language can lead to pauses or unexpected slowing of program execution. Research into the field of garbage collection in an attempt to apply deterministic garbage collection algorithms to the JVM may prove fruitful.[2]

Clearly, there are a number of possible areas of future research that lead on from our work. Research into these problems may help realize the elegant approach taken by JAIN. In this fashion, telecommunications services will gain the benefits of portability without losing any of the highly desirable existing characteristics such as fault tolerance, rapid development and high performance.

[2]In C or C++ a developer may defer memory collection operations until a performance critical section of code has finished executing. In Java the developer has little control over memory operations. The Java.System.gc() method is able to hint to the garbage collector that it should clean up. This is only a hint and a garbage collector may ignore the hint.

Appendix A

The Implementation of Internet Call Waiting

A.1 Introduction

The Internet Call Waiting service described in the earlier sections has been implemented as a prototype. It is written in the Java programming language. Other than the use of the JAIN TCAP API the service prototype does not rely on any *container structures* or frameworks that are not part of the Java 2 platform standard edition such as Enterprise Java-Beans or Java-Spaces.

A.2 Overview

The prototype simulates portions of the AIN and implements the ISP's ICW server and the call recipient part. Separate application processes simulate the telephone devices, SSP and SCP. The SSP and SCP simulators communicate using the JAIN TCAP API. The implementation of the JAIN TCAP API used for the simulation runs on an Ericsson-Infotech dual ported physical interface. The ports on the physical interface are physically connected to each other via a V.35 cable. Two instances of an SS7 stack execute and each is configured to use a port on the physical interface. These stacks and their corresponding physical interface reside on a Solaris 2.6 host. As there is only one physical interface the SSP and SCP must run on the same host. Telephone devices send data (such as the dialed digits) to the SSP using TCP/IP.

The ISP's ICW server is implemented as an application process. This application communicates with the SCP via TCP/IP. As these processes communicate using TCP they may execute on different hosts. The call recipient part software is a Java application that receives call notification messages via a TCP/IP connection. Serialized Java objects are passed over all TCP connections for the sake of reducing parsing complexity.

Communication between these processes is shown in Figure A.1.

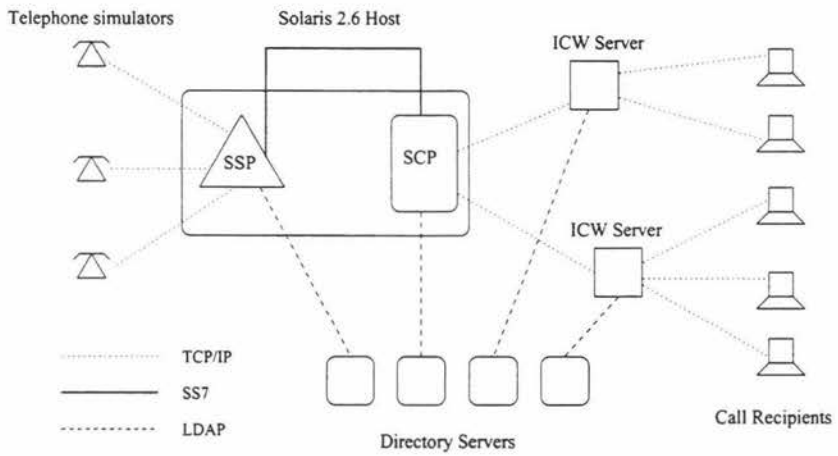

Figure A.1: Communication between the simulators. Both the SSP and SCP reside on the same host. The Status Monitor is not included for clarity.

LDAP – lightweight directory access protocol – directory servers are used by the SSP, SCP and ISP's ICW server. The SSP uses the LDAP server to store the list of active calls. The SCP uses the LDAP server to look up both the caller name - caller number pairs and the ISP's 800 access number - ICW server IP address pairs. The LDAP server is accessed through the Java Naming and Directory interface (Sun Microsystems, 1999f). The service may be demonstrated without a directory server present by hard-coding these values into a Java Map class.

The configurable parameters of each of these simulators is read from configuration files during the construction of the objects. These configuration files are *Properties* files parsed by the *java.util.Properties* class contained in

the Java Development Kit and Run-time Environment.

A.3 Displaying the prototype

In order to demonstrate the prototype clearly an extra process exists. This is termed the *Status Monitor.* This process receives data from the SSP, SCP and ISPs via TCP/IP. When data arrives from these processes it is parsed and displayed in a number of panes within the display window.

Figure A.2 shows the Status Monitor displaying call information for three calls. The first call is to a non busy number. Hence that call is not applicable to the ICW service. This is shown in the SSP pane by the text "Dialed number is not busy; call not a candidate for ICW". In the second call the dialed party is busy and not connected to an ISP. The SSP queries the SCP to find if the dialed number is not a candidate for ICW. Both the SSP and SCP display information on this call as they are both involved in the processing of the particular call. The SSP displays that 81110032 is involved in a call with 8110030. The SSP contacts the SCP and displays the result that the dialed number is not a candidate for ICW. This is the second grouping of text in the SSP pane.

The SCP determines that the dialed number is not an ISP's 800 access number. This information is displayed in the first line of the SCP's pane

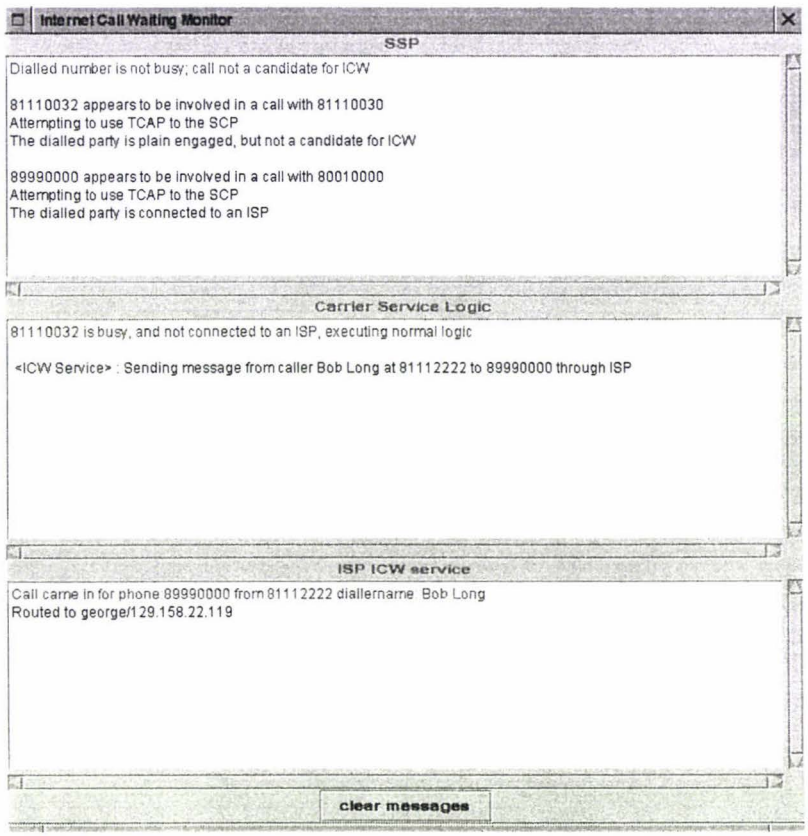

Figure A.2: The Status Monitor.

(The SCP's pane is titled "Carrier Service Logic").

In the third call the dialed party is connected to an ISP. The dialed party has also subscribed to the ICW service. There are three messages shown in the Status Monitor during this call. The first is the SSP querying the SCP to determine if the called party is connected to an ISP. The second message is from the SCP determining that the dialed party is indeed connected to an

ISP and a call notification message is sent to the ISP. The third message is displayed in the ISP's pane. It states that the ISP's ICW server has received a call notification message and has sent an appropriate message to the call recipient. The call recipient is located on host-name "george" with the IP address 129.158.22.119.

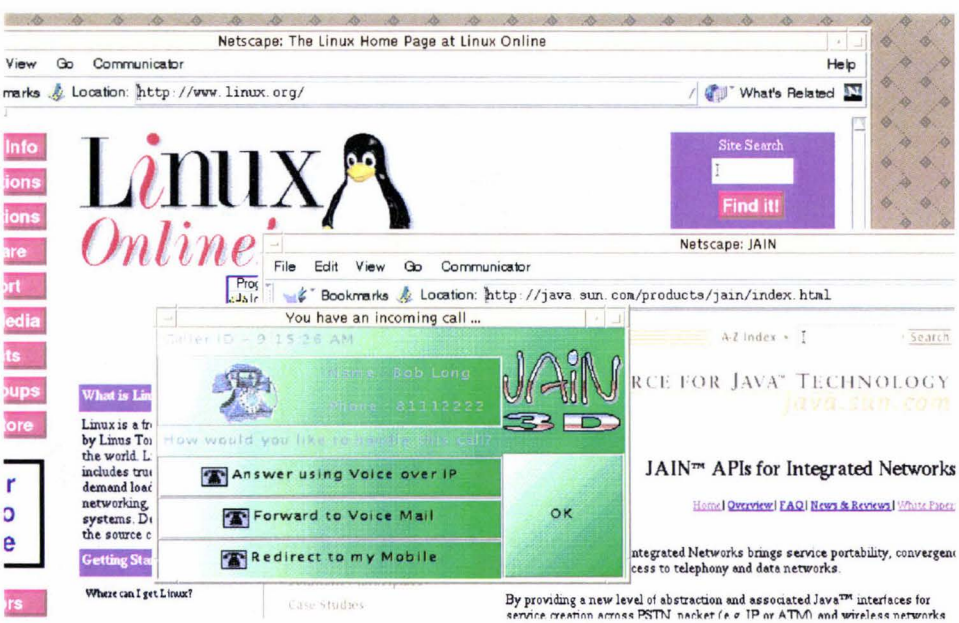

Figure A.3: An incoming call notification.

A screen capture of the call recipient part displaying an incoming call notification is shown in Figure A.3. This picture shows the called party software receiving the call notification. The call being received is the same as the last call used to demonstrate the Status Monitor. It can be seen that

both the calling party's name and number are displayed in the window. This picture is from a Solaris 2.6 host.

Appendix B

Glossary

Acronym	Expansion
AIN	Advanced Intelligent Network
ANSI	American National Standards Institute
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
CTD	Click-to-Dial
DPE	Distributed Processing Environment
GLP	Gateway Location Protocol
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
ICW	Internet Call Waiting
IETF	Internet Engineering Task Force
IN	Intelligent Network
INAP	Intelligent Network Applications Part
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISO	International Standards Organisation
ISP	Internet Service Provider
ISUP	ISDN User Part
ITU	International Telecommunications Union
ITU-T	International Telecommunications Union
JAIN	A set of Java API's for Integrated Networks
JMF	Java Media Framework
JNDI	Java Naming and Directory Interface

Acronym	Expansion
JNI	Java Native Interface
LDAP	Lightweight Directory Access Protocol
MGCP	Media Gateway Control Protocol
MTP	Message Transfer Part
ORB	Object Request Broker
OSI	Open Systems Interconnect
PEP	Protocol Extensions Protocol
PIN	Personal Identification Number
PSTN	Public Services Telephone Network
RGW	Residential Gateway
RMI	Remote Method Invocation
RTCP	Real Time Control Protocol
RTP	Real Time Transport Protocol
RTSP	Real Time Streaming Protocol
SCCP	Signaling Connection Control Part
SCE	Service Creation Environment
SCP	Service/Signal Control Point
SDP	Session Description Protocol
SIB	Service Independent Building Block
SIP	Session Initiation Protocol
SLEE	Service Logic Execution Environment
SMTP	Simple Message Transfer Protocol
SPC	Stored Program Control
SS7	Common Channel Signaling System Number Seven
SSN	Sub System Number
SSP	Service/Signal Switching Point
STP	Service/Signal Transfer Point
TCAP	Transactional Capabilities Application Part
TCP	Transfer Control Protocol
TGW	Trunking Gateway
TINA	Telecommunications Information Networking Architecture
TTCN	Tree and Tabular Combined Notation
TUP	Telephone User Part
UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Universal Resource Locator
VoIP	Voice over IP

Bibliography

Anjum, F., Caruso, F., Jain, R., Missier, P., & Zordan, A. (1999). *Chaitime:* A System for rapid creation of portable next generation telephony services using third-party software components. In *IEEE Conference on Open Architectures and Network Programming (OPENARCH)*, pp. 22–31 New York.

ANSI (1995). Signaling system no. 7 (ss7) - broadband integrated services digital network user part (B-ISUP). Tech. rep., American National Standards Association, ftp://ftp.t1.org/pub/t1stds/648-95.txt. ANSI T1.648-1995.

Bellcore (1991). Advanced Intelligent Network (ain) Release 1, Switching Systems Generic Requirements. Tech. rep., Bellcore.

Braden, R., Zhang, L., Berson, S., Herzog, S., & Jamin, S. (1997). Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. Tech. rep., Internet Engineering Task Force. IETF RFC 2205.

Cuervo, F., Hill, B., Green, N., Huitema, C., Rayan, A., Rosen, B., & Segers, J. (2000). Megoco Protocol. IETF Internet Draft.

Fielding, R., Gettys, J., Mogul, Nielsen, H., & Berners-Lee, T. (1997). Hypertext Transfer Protocol - HTTP/1.1. Tech. rep., Internet Engineering Task Force, http://www.ietf.org/rfcs/rfc2068.txt. IETF RFC 2068.

Gosling, J., Joy, B., & Steele, G. (1999). *The Java Language Specification.* Addison-Wesley.

Handley, M., & Jacobsen, V. (1998). SDP: Session Description Protcol. Tech. rep., Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2327.txt. IETF RFC 2327.

Handley, M., Schulzrinne, H., Schooler, E., & Rosenberg, J. (1999). SIP: Session Initiation Protocol. Tech. rep., Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2543.txt. IETF RFC 2543.

Huitema, C., Cameron, J., Mouchtaris, P., & Smyk, D. (1999). An Architecture for Residential Internet Telephony Service. *IEEE Network Magazine*, *13*(3), 50–56.

ISO/IEC (1997). *9646: Part 3: The Tree and Tabular Combined Notation (TTCN)* (Second edition). ISO/IEC.

ITU-T (1992). Recommendation Q.1201 Principles of Intelligent Network Architecture. Tech. rep., International Telecommunications Union.

ITU-T (1995a). Recommendation Q.1213: Global Functional Plane for Intelligent Network CS-1. Tech. rep., ITU-T.

ITU-T (1995b). Recommendation Q.2761: Broadband Integrated Services Digital Network (b-isdn) - Functional description of the B-ISDN user part (B-ISUP) of signalling system no.7. Tech. rep., Internation Telecommunications Union.

ITU-T (1997a). Recommendation H.323. In *Packet-Based Multimedia Communication Systems, Version 2.* International Telecommunications Union.

ITU-T (1997b). Recommendation X.680: Abstract Syntax Notation One (ASN.1): Specification of basic notation. Tech. rep., ITU, http://www.itu.int/itudoc/itu-t/rec/x/x500up/x680.html.

Keck, D. O., & Kuehn, P. J. (1998). The Feature and Service Interaction Problem in Telecommunications Systems: A Survey. *IEEE Transactions on Software Engineering*, *24*(10), 779–796.

Kozik, J., Montgomery, W. A., & Stanaway, J. J. (1998). Voice Services in Next-Generation Networks: The Evolution of the Intelligent Network and Its Role in Generating New Revenue Opportunities. *Bell Labs Technical Journal*, *3*(4), 124–143.

Kuepper, A. (1999). Locating TINA User Agents: Strategies for a Broker Federation and Their Comparison. In *6th International Conference on Intelligence in Services and Networks*, pp. 416–428.

Lewis, D., & Tiropanis, T. (1998). Integrating TINA into an Internet-based Services Market. In *5th International Conference on Intelligence in Services and Networks*, pp. 183–192.

Lombardo, A., Nicosia, P., Palazzo, S., Samarotto, M., & Schembra, G. (1999). Performance Evaluation of an Allocation Strategy for TINA Compliant Mobile Agents Supporting PCS in a Multi-retailer Environment. In *6th International Conference on Intelligence in Services and Networks*, pp. 401–415.

McKinney, R. D., Montgomery, W. A., Ouibrahim, H., Suben, P., & Stanaway, J. J. (1998). Service Centric Networks. *Bell Labs Technical Journal*, *3*(3), 98–115.

OMG (1997). The Common Object Request Broker Architecture. Tech. rep., Object Modelling Group. Version 2.1, Available from http://www.omg.org.

Rizzetto, D., & Catania, C. (1999). A Voice over IP Service Architecture for Integrated Communications. *IEEE Network Magazine*, *13*(3), 34–40.

Rosenberg, J., Lennox, J., & Schulzrinne, H. (1999). Programming Internet Telephony Services. *IEEE Network Magazine*, *13*(3), 42–49.

Rosenberg, J., Salama, H., & Squire, M. (2000). Telephony Routing over IP (TRIP). IETF Internet Draft.

Rosenberg, J., & Schulzrinne, H. (1999). A Framework for Telephony Routing over IP. IETF Internet draft.

Schieferdecker, I., Li, M., & Hoffman, A. (1998). Conformance Testing of TINA Service Components - The TTCN/CORBA Gateway. In *5th International Conference on Intelligence in Services and Networks*, pp. 393–408.

Schneier, B. (1995). *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (Second edition). John Wiley and Sons.

Schulzrinne, H., Rao, A., & Lanphier, R. (1998). Real Time Streaming Protocol (RTSP). Tech. rep., Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2326.txt. IETF RFC 2326.

Schulzrinne, H., Cassner, S., Frederick, R., & Jacobsen, V. (1996). RTP: A Transport Protocol for Real-Time Applications. Tech. rep., IETF, http://www.ietf.org/rfc/rfc1889.txt. IETF RFC 1889.

Schulzrinne, H., & Rosenberg, J. (1998). A Comparison of SIP and H.323 for Internet Telephony. In *Proceedings International Workshop on Network and Operating Systems support for Digital Audio and Video(NOSSDAV'98)*.

Schulzrinne, H., & Rosenberg, J. (1999). The IETF Internet Telephony Architecture and Protocols. *IEEE Network Magazine*, *13*(3), 18–23.

Stamoulis, G., Kalopsikakis, D., Kirikoglou, A., Siris, V., Prevedourou, D., Efremidis, S., & Jormakka, H. (1999). Agent-Based Service and Retailer Selection in a Personal Mobility Context. In *6th International Conference on Intelligence in Services and Networks*, pp. 429–442.

Stefani, J.-B., Dumant, B., Tran, F. D., & Horn, F. (1998). The ReTINA DPE Kernel: A Flexible, Real-Time ORB Framework. In *5th International Conference on Intelligence in Services and Networks*, pp. 289–296.

Sun Microsystems (1997). Java Native Interface Specification. Tech. rep., Sun Microsystems, http://java.sun.com/products/jdk/1.3/-docs/guide/jni/spec/jniTOC.doc.html.

Sun Microsystems (1999a). JAIN APIs for Integrated Networks. Tech. rep., Sun Microsystems, http://java.sun.com/products/jain/.

Sun Microsystems (1999b). JAIN: Integrated Network APIs for the Java Platform. Tech. rep., Sun Microsystems, http://java.sun.com/-products/jain/jain_wpfinal.pdf.

Sun Microsystems (1999c). Jain TCAP API: Public Draft. Tech. rep., Sun Microsystems, http://java.sun.com/aboutJava/communityprocess/-review/jsr011/index.html.

Sun Microsystems (1999d). JAIN(tm) SS7 OAM API: Public Draft. Tech. rep., Sun Microsystems, http://java.sun.com/aboutJava/-communityprocess/review/jsr018/index.html.

Sun Microsystems (1999e). Java Media Framework. Tech. rep., Sun Microsystems, http://java.sun.com/java-media/jmf/index.html.

Sun Microsystems (1999f). Java Naming and Directory Interface (JNDI). Tech. rep., Sun Microsystems, http://java.sun.com/products/jndi.

Sun Microsystems (1999g). Java(tm) Cryptography Extension 1.2. Tech. rep., Sun Microsystems, http://java.sun.com/products/jce.

Sun Microsystems (1999h). Java(tm) Telephony API Home Page. Tech. rep., Sun Microsystems, http://java.sun.com/products/jtapi.

The Parlay Group (1999). Parlay Specification 1.2. Tech. rep., The Parlay Group, http://parlay.msftlabs.com/Papers/parlayAPI_1_2.zip.

Tiropanis, T. (1998). Offering Role Mobility in a TINA Environment. In *5th International Conference on Intelligence in Services and Networks*, pp. 89–100.

Wind, B., Samarotto, M., Nicosia, P., Lambrou, M., & Tzifa, E. (1998). Enhancing the TINA Architectural Framework for Personal Mobility Support. In *5th International Conference on Intelligence in Services and Networks*, pp. 233–248.