University of Namur

MASTER IN COMPUTER SCIENCE

Network convergence for the provision of in services in the Internet

Nicoll, Stephane

Award date:
2001

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique
Année Académique 2000-2001

NETWORK CONVERGENCE FOR THE PROVISION OF IN SERVICES IN THE INTERNET

Stéphane Nicoll

Promoteur : Olivier Bonaventure

Mémoire en vue de l'obtention du grade de Maître en Informatique

Abstract

Networks convergence is now a very discussed subject. The interaction between the Internet and traditional networks announces the emergence of a set of services which are to be very attractive for the end-user.

Mechanisms offering advanced telephony systems on the Internet are analysed in this dissertation suggesting different solutions to be applied nowadays: on the one hand those involving the use of both networks – mainly the approaches scrutinised by the IETF – and on the other hand the implementation of higher-level systems resulting from an independent consortium and allowing the abstraction of the underlying network.

Those different approaches are illustrated by a key-example in which the reader is offered the possibility of comparing the strong and weak points of each approach in an efficient way.

Keywords: Internet telephony, IN, SIP, PINT, SPIRITS, Parlay, JAIN.

Résumé

La convergence des réseaux est actuellement un sujet très discuté. L'interaction entre l'Internet et les réseaux traditionnels promet la venue d'un ensemble de services très attrayants pour l'utilisateur.

Ce mémoire étudie des mécanismes permettant de fournir des services de téléphonie avancés sur l'Internet ; il propose de dégager certaines des différentes solutions qui sont développées actuellement : d'une part celles – principalement étudiées par l'IETF – qui impliquent l'utilisation des deux réseaux et d'autre part l'installation de systèmes de plus haut niveau résultant d'un consortium indépendant et permettant d'abstraire complètement le réseau sous-jacent.

Ces différentes solutions sont illustrées par un exemple unique permettant de comparer efficacement les points forts et les points faibles de chaque approche.

Mots-clés : Internet telephony, IN, SIP, PINT, SPIRITS, Parlay, JAIN.

Acknowledgements

Whatever the author of a report, acknowledgements are for me to be considered with care. No matter if you agree or disagree with my point of view, I sincerely hope you will read them carefully.

First I would like to thank my training course master, Renaud Meurisse. All my studies I was in search of a model, a reference as a computer specialist; someone we can admire for both his professional and human qualities. Unfortunately I experienced many disappointments in my search for it. Renaud is one of the very best computer specialists I have ever known and also an excellent colleague. His patience has stood the test of time and his expertise is to be recognised and appreciated. I would like to express my deepest gratitude for his very helpful hand during my work placement with Alcatel. I would also like to thank the Alcatel staff who offered me a real picture – sometimes even disruptive – of professional life. Living such a situation fully and feeling oneself involved in the process gives a total new light and dimension on the professional activity I used to imagine. Beside Renaud, I would like to thank especially Eric Delmelle, François Goffin, Nathalie Heilly, Jean-Michel Hemstedt and Jacques Meirlan for their full assistance and support.

At the University of Namur, I also acknowledge my debt to Steve Uhlig for his total devotion, sometimes even during extra hours. Steve is the most crazy computer scientist I have ever met even though such crazy guys do not lack in such a beautiful field as computer science. I hesitate to wish him to quench his thirst of research, his insatiable need of knowledge since he feels at ease in his domain. I wish him all the best for the future.

My thanks also to the Internet community which delivers us information and resources in such an easy way. We always hear about the negative aspects of the Internet, its faults and its vices. Much should be done regarding that aspect but this should not let us forget all that the Internet offers as benefits: information, knowledge, help and assistance.

I guess I could not thank all the people who have helped me in my research, either technically or morally. I do hope they will recognise themselves. I also would like to express my gratitude to Peter Kelly, Vincent Lobet and Marvin Jacobson for their linguistic supports. My thanks are extended to all the readers who read my drafts and provided me with a constructive and critical feedback. My warmest thanks to Jean-Michel Hemstedt and Steve Uhlig in that respect.

This research, which completes a 5-year-course, is the best opportunity for me to thank my parents for all they have done for me since I was born. As many other persons I have not devoted much time to them. Yet, as they have taught me I will still look forward hoping they will forgive me in the very near future. Thanks also to Vanessa for her discrete and efficient help during the numerous efforts linked to the writing of this report. As usual she could separate priorities and unessential matters but leaving me to my own responsibilities for the final decision. Her patience and her broad-minded touch will always astonish me.

Last but not least, my sincerest thanks to my supervisor, Olivier Bonaventure, for his determined help, which I often misinterpreted. In spite of our divergences, I will keep the greatest respect and satisfaction to have been honoured to work with him.

A la mémoire des proches qui sont partis
Vous me manquerez à jamais.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this introduction we invite the reader to analyse the title of our dissertation briefly. The latter seems very rich to our eyes and contains a lot of information allowing to define the limits of the scope of this dissertation. We will end up the introduction by explicating the logic used when writing the dissertation and the different chapters included.

1.1 Short analysis of the title

Nowadays two categories of networks exist. First, telecommunication networks or traditional networks. Those networks refer to the whole of the infrastructure used -mainly- to deliver telephony services. The second category is made of data networks of which the Internet is the most well-known. A tendency in fashion at the present time is to bring those two types of networks closer, and this, for several reasons. The first one results from an increasing problem taking place for the past few years: the importance of data traffic in comparison with "normal traffic". As we have specified, traditional networks were designed to deliver telephony services and thus *a fortiori* does not resist to increasing data traffic. Figure 1-1 illustrates this issue.

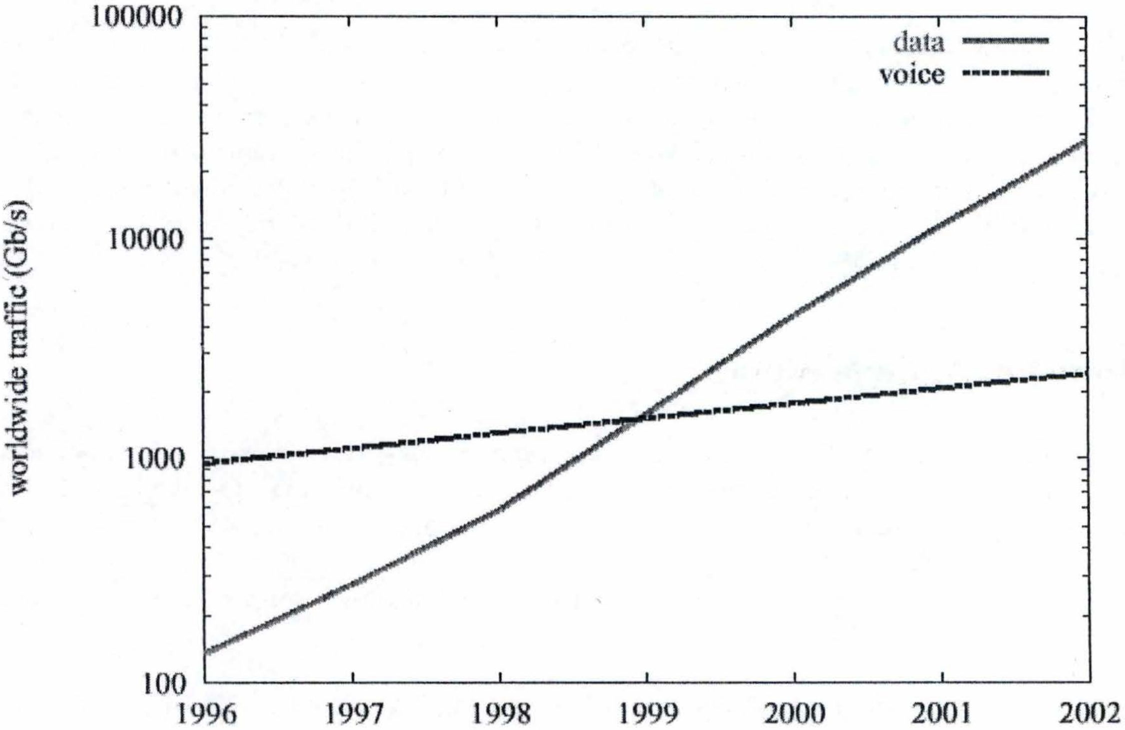


Figure 1.1 – Voice vs. data traffic

The second reason is the consequence of an increasing interest for the use of new technologies proceeding from the Internet, and this with a view to permitting the transit of telephonic traffic. Beside the performance aspect, this choice allows a general cost reduction. All those reasons have created the interest to transporting telephony traffic throughout networks initially designed for data traffic transit. This interest is on the basis of the "Internet Telephony" concept, which to be viable has to allow the interaction between the data network and the telephony network. The term "hybrid network" will be used to refer to the whole of the two networks and the whole will constitute the term "network convergence", i.e. the title of this dissertation.

An important point, which is on the basis of this dissertation, is to take into account the supplementary services linked to telephony. Those services are developed thanks to the assistance of intelligent networks (INs) prevailing in the telephony network. Introducing new services before their existence was, indeed, a complex, costly and long process. The IN reign is linked to its potentiality to separate the "logic service" from the inferior levels allowing the traffic itself. Moreover, this concept provides both a framework and a set of tools allowing the rapid development and installation of services in the network. The term "IN services" in the present dissertation refers, in this respect, to those services and, as we will see, to new services proceeding from the use of hybrid networks. As a matter of fact, our dissertation will follow a twofold approach relating to services. Firstly, it will see to it that the mechanisms permitting traditional services support on the hybrid network are identified and secondly the dissertation will indicate how new services can be delivered.

In this dissertation, when referring to "deliver", the term "provision" is to be understood in its broadest definition, i.e. including development, installation, support and evolution of services. In reality, we use the term "provision" to refer to the whole process to be followed to deliver a service to the user.

Finally, the term "Internet" is very controversial. As the title sounds we concentrate on the delivery of IN services in the Internet (more specifically to the essential solutions needed for services delivery on an hybrid network). Yet, it is necessary to clarify that this convergence strives towards an objective called Next Generation Network or NGN. This ultimate network would consist of a complete merging between the two types of networks described supra and is, therefore, the final step of the convergence. Although we will limit the scope of this dissertation to an hybrid system, the NGN concept will hang over the whole dissertation.

1.2 Objective and contents

The objective of the present dissertation is to study the different means allowing services delivery on an hybrid network. Our work is neither economists nor network managers-oriented. The aim is rather to single out the different steps linked to convergence in a service-designed approach. This justifies why -for both solutions- we have not gone into technical details deeply. Considering the extreme novelty of the matter, it appeared to us that drawing an overall view was more pertinent.

Conceptually, the dissertation is structured in three parts. The first part describes the existing and present situation with no convergence interference. The second one outlines the technical aspects allowing convergence achievement. Finally, in the last part we will study the means allowing the delivery of services, notwithstanding if they are new or not.

Chapter 2 offers a relatively broad overview both of intelligent networks and of the way of how they are developed and supported at the present time. Appendix 4 describes the signalling system protocol in use in traditional networks and illustrates this chapter.

Chapter 3 explains the mechanisms involved in the creation of an hybrid network. This creation involves the birth of entities on the borderline between networks allowing the interaction between the data network and the voice network. Once the mechanisms linked to this architecture having been explicated, the chapter presents a generic model allowing service delivery with this architecture. Appendix 5 describes the signalling system protocols in the Internet and can be considered as a counterpart of Appendix 4. More precisely, it describes two protocols and chooses one of the two.

The last three chapters constitute the last part of the dissertation. After their introduction in chapter 3, those chapters use a well-known common service: the "call center" or "helpdesk". This service offers a helping service delivery to the customers. Traditionally, intelligent networks are used in two ways for this type of service. Firstly, the access to a call center is generally operated via a free call number (a 800 type). The first task for the IN is to assign a call center related number based either on information linked to the context (time, day, etc.) or on the calling number (calling number area, phone number, etc.). The second task is to design the operator who either has to answer the call or retrieve extra information in order to guide the client to the appropriate service.

Developing an hybrid network allows the introduction of a set of new possibilities. Traditional services, indeed, can be customised, ameliorated when they interact with email, web and directories services options. The development of this last part will bring us closer and closer to NGN. Chapter 4 presents purely hybrid solutions in the sense that they have been designed, chosen to be used for both the traditional and the voice networks. Chapter 5 is near to the NGN philosophy as it describes open and programmable networks. Those networks are not hybrid any more and can provide services for traditional network users too. Those two chapters make up the theoretical approach and are largely illustrated by the call center example. Finally, chapter 6 discusses the pre-implementation of such a network. We cannot speak of an implementation since the objective of the work was to prepare and study the environment in which the implementation itself was to take place.
Conclusions of the present dissertation are to be found in chapter 7.

Chapter 2

Overview of Intelligent Networks

The purpose of this chapter is to get the reader used to Intelligent Network (IN) concepts. We subdivide it into 5 parts. First of all, we introduce briefly the functioning of a telecommunication network. Secondly, we provide a short history of INs. Thirdly, definition, objectives and properties are given. We go on, in the fourth section, with the study of the IN architecture and we conclude this chapter with examples of IN services.

2.1 Introduction

This introduction presents basic concepts of telecommunication networks in general, and telephone networks in particular. It is mainly based on [TANE96, pp. 102-155] and [FGL00, pp. 21-25]. The telephone networks of today offer an integrated voice and data services to the end user. Very roughly, data services are provided by the Integrated Services Digital Network (ISDN), while the Public Switch Telephone Network (PSTN) is more voice-oriented. In the remainder of this dissertation, we consider the PSTN in its most general sense, i.e. with the ISDN infrastructure. Effectively, the ISDN is a technology supported within the PSTN, not a separate network.

When telephone networks first came into being, terminals were sold by pairs since the two telephones involved in a call must be connected directly. Then the notion of a switching office was born. This notion of switching has evolved from manual systems managed by human operators to automatic systems (mechanically first and digitally afterwards). Connecting every switching office to every other switching office became quickly unmanageable. This is the reason why a hierarchy in the different switching offices was established (up to five levels)[1].

There are four main components in the PSTN structure. First of all, the Customer Premise Equipment (CPE) can be a simple terminal but also a complex enterprise Private Branch Exchange (PBX) system. Its primary functions are to transmit and receive signals between the customer and the network. Secondly, the transmission facilities provide the communication paths and equipments to amplify or regenerate signals. Thirdly, switching systems interconnect the transmission facilities and route the traffic through the network. Finally, Operations Administration and Management (OA&M) systems provide useful functions to the network operator. Unfortunately, it is difficult to find a worldwide accepted standardisation for the transmission facilities and switching systems. The wire connection between the customer's CPE and the nearest end office (also called the local exchange, the "local switch") is called the local loop. We bring up the hierarchy notion again: the end office is generally a class 5 switch. Each end office has a number of outgoing lines to one or more nearby switching centres, called tool offices (or tandem offices if they are in the same area which is constituted by the prefix of phone numbers). These components are generally class-4 switches and are connected to bigger switching centres. The reader could refer to [TANE96, pp. 134-135] for more details.

[1] This hierarchy has been implemented by AT&T but other companies and countries use the same general principles.

While the communication in the local loop is usually analog[1], the core network is digital nowadays. This has some advantages over analog, the most important of which is to transform this network into a multi-media one. Figure 2.1 shows the components involved in a relatively long-distance call (e.g. the parties are not connected to the same local exchange).

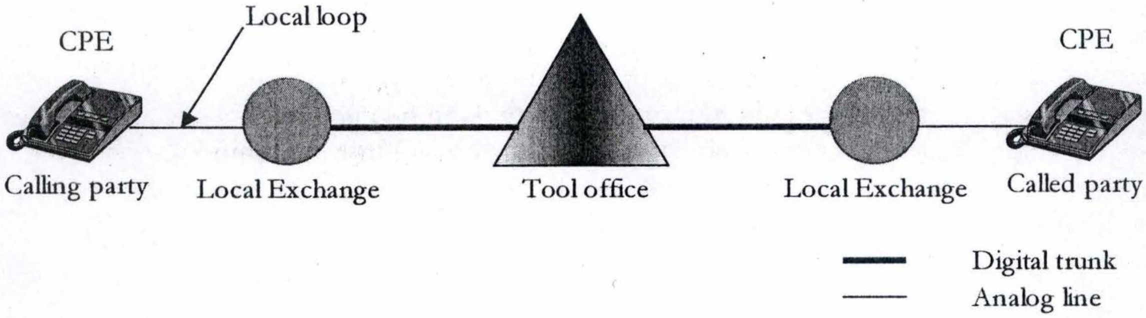


Figure 2.1 – Main components involved in a relatively long-distance call

Each end office has a codec that converts analog signals into digital ones and vice versa. When the CPE is a computer, modems perform roughly the same conversion as this codec. Indeed, the purpose of the modem (modulator-demodulator) is to transport digital data over analog circuits. This is not trivial since many problems may occur, such as attenuation, delay, distortion and noise on analog circuits. [TANE96, pp. 109-115] gives a good explanation of these problems and how modems handle them.

We will end this section by giving some explanations of the way voice is transported over these networks. Unlike IP networks, the voice is transmitted over circuits. Since the cost of installing a trunk between two switching offices is not bandwidth-dependent, telephone companies have developed complex schemes to multiplex as many conversations as possible over a single physical trunk. These can be classified into two groups: Frequency Division Multiplexing (FDM) and Time Division Multiplexing (TDM), which is the most widely used. In Frequency Division Multiplexing, each logical channel has its own frequency band and has the exclusive use of it.

In Time Division Multiplexing, the user periodically takes the entire bandwidth for a little burst of time. The scheduling of users is performed by a round robin scheme. TDM can be handled entirely by digital equipment (unlike FDM) and so is far more widely used. However, a conversion between analog and digital signals is needed in end offices, as shown in Figure 2.1. This conversion is performed by the Pulse Code Modulation (PCM) codec. It produces a 7- or 8-bits sample every 125 µsec, i.e. 8000 samples per second (i.e. 64kbps). This is driven by the need to respect the Nyquist theorem which states: "any signal can be digitised by sampling provided that exact samples are taken at least at twice the highest frequency of the signal". Since telephone networks have roughly a 4khz channel bandwidth, this theorem is respected. These samples are transported over high-bandwidth lines. A well known one is the T1 carrier, which is constituted by 24 voice channels multiplexed together. In this scheme, a 193 bits-frame is sent every 125 µsec (the extra bit is used for framing). Another PCM carrier is the E1, which has 32 voice channels. T1 has a bandwidth of 1.544 Mbps and is widely used in North America and Japan while E1 has a bandwidth of 2.048 Mbps and is mainly used in Europe. It should be noted that these carriers could be multiplexed in higher-level carriers (for example, four T1 frames can be multiplexed into a T2 frame for a total bandwidth of 6.312 Mbps).

[1] The cost of converting local loops in digital lines (e.g. ISDN) is enormous.

2.2 History of Intelligent Networks

In this section, we will see how IN has evolved from a service which is switch-dependent to a more sophisticated network where intelligence is taken out of switches into special devices. This history of IN is mainly based on [TELECO]. Plain Old Telephone Services (POTS) are the very first approach to what is going to be the IN concept[1]. In POTS, the intelligence is located in the switch and this results in many drawbacks: slow deployment, no extensibility at all to services and heterogeneous networks. In fact, when a network operator wants the deployment of a service, (s)he provides specification to the switch vendor who builds a specific device for this service. This implies that a customer in an area has not necessarily the same service as another customer in another area. Some features delivered by POTS are, for example, call waiting and call forwarding.

A great step forward in the development of services is achieved by Stored Program Control (SPC). Indeed, in SPC the service is programmable where in POTS it is hardwired: this allows new services to be added more easily. However, this evolution continues to be specific from a service logic point of view and thus the addition of new services is increasingly hard because of this dependence between the switch and the service-specific logic. It implies that the service logic used in one service cannot be used in another service. More modularity and reusability of components must be achieved.

The next evolution is the emergence of a powerful signalling system that is still used today called Common Channel Signalling System number 7 (SS7). The user parts of SS7 offer useful functionalities. The knowledge of SS7 is not critical for the good understanding of this dissertation. However, the interested reader may consult the fourth appendix devoted to SS7 in particular and to signalling protocol in telephone networks in general. Without anticipating the content of this appendix, we may note that SS7 allows the introduction of new services, such as Caller ID who provides the calling party's telephone number to the called party, which is transmitted over the SS7 network.

It is in this environment that the first version of IN was born (Figure 2.2). For the first time, service-logic is external to switches in a special hardware called Service Control Point (SCP).

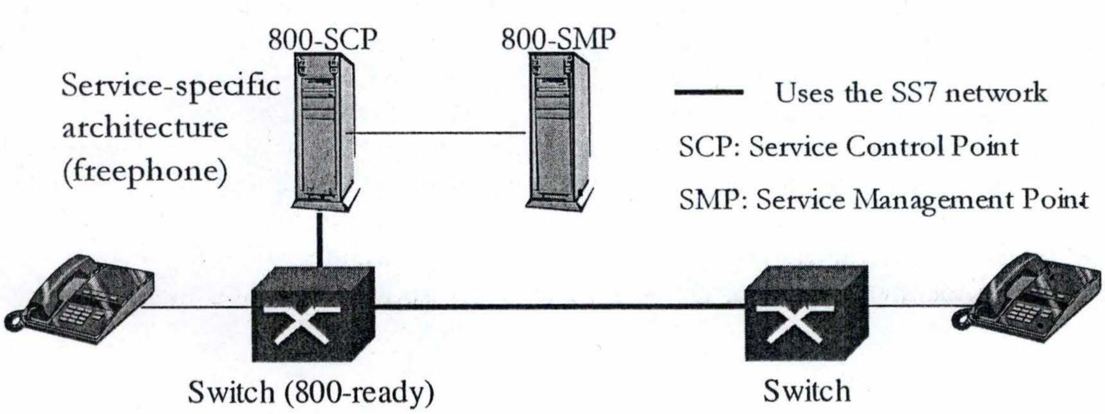


Figure 2.2 – IN/1 architecture

[1] Whether POTS are the first approach of IN or not may vary. Some are saying that POTS only allow a phone call to be made and it is the PSTN that add extra features to the network.

Of course, communication is needed between the old system and the SCPs: software is developed in the switches and uses SS7 networks. The aim of this software is to detect whether the call was an "IN call" or not. Triggers are associated with each service and address a specific SCP. New services, such as freephone (800), can be developed with this new architecture. Unfortunately, as shown in the figure above, although the service-logic is external to the switching system, it is still service specific. For example, a 800-service has a 800-trigger at the switching system, a 800-service database at the SCP and a 800-Service Management Point (SMP) to support the 800-SCP. Thus the 800-service set of capabilities cannot be used for other services. This major drawback is resolved in Advanced Intelligent Network (AIN) that introduces the first service-independent architecture, in which a given part of a telephone number can be interpreted differently by different services depending on factors such as time of day, caller identity and type of call. The concept of AIN[1] has had great success and is the basis of the IN architecture used today in many telecommunication networks.

2.3 Intelligent Network: definition, actors and objectives

An Intelligent Network can be defined as a service-independent architecture that allows the provision and operation of new services rapidly and efficiently. We have seen in the previous section that the service logic handling a call is located separately from the switching facilities. This allows services to be added or changed without having to redesign switching equipment.

There are four main actors in the IN scene. The first, the service user, is the end-user of the service. For example, this is the person who calls a free-phone or utilizes his calling card to call a friend while he is abroad. The second, the service subscriber, is the actor who subscribed to an IN feature. He can use it for himself or provide it to his customers. The third, the service provider, creates, deploys and supports IN services. He has contracts with service subscribers. These contracts specify the billing and the subscribed features. The last one, the network operator, provides the IN infrastructure needed to support IN services. He has contracts with service providers.

The service-independence is the main property of an IN: the history of IN has shown that service-independence drives all other requirements. IN must facilitate the addition of new services whatever the service/network implementation is. It must function in a multi-vendor environment and can allow the inclusion of additional capabilities to the network. This independence can be approached in two ways. The first way, i.e. the service way, allows service providers to define their own services independently of service specific developments by equipment vendors. With this independence, the network infrastructures can evolve without affecting existing services. The second way, i.e. the network way, allows network operators to allocate functionalities and resources within their networks and to efficiently manage their networks independently of network implementations and specific developments by equipment vendors.

Intelligent Networks are characterised by many fields (see [ALCA00a]). We give 3 of them. The first is the development time that must be as fast as possible. Secondly, the possibility for both service subscribers and service providers to customize their services. Then the portability of services. However, this vision of creating a device-independent network in which services are

[1] AIN is rather an American concept. The ITU has developed an equivalent version of AIN called Capability Set 1 (CS1). It comes in evolutionary subsets called the Core INAP capabilities.

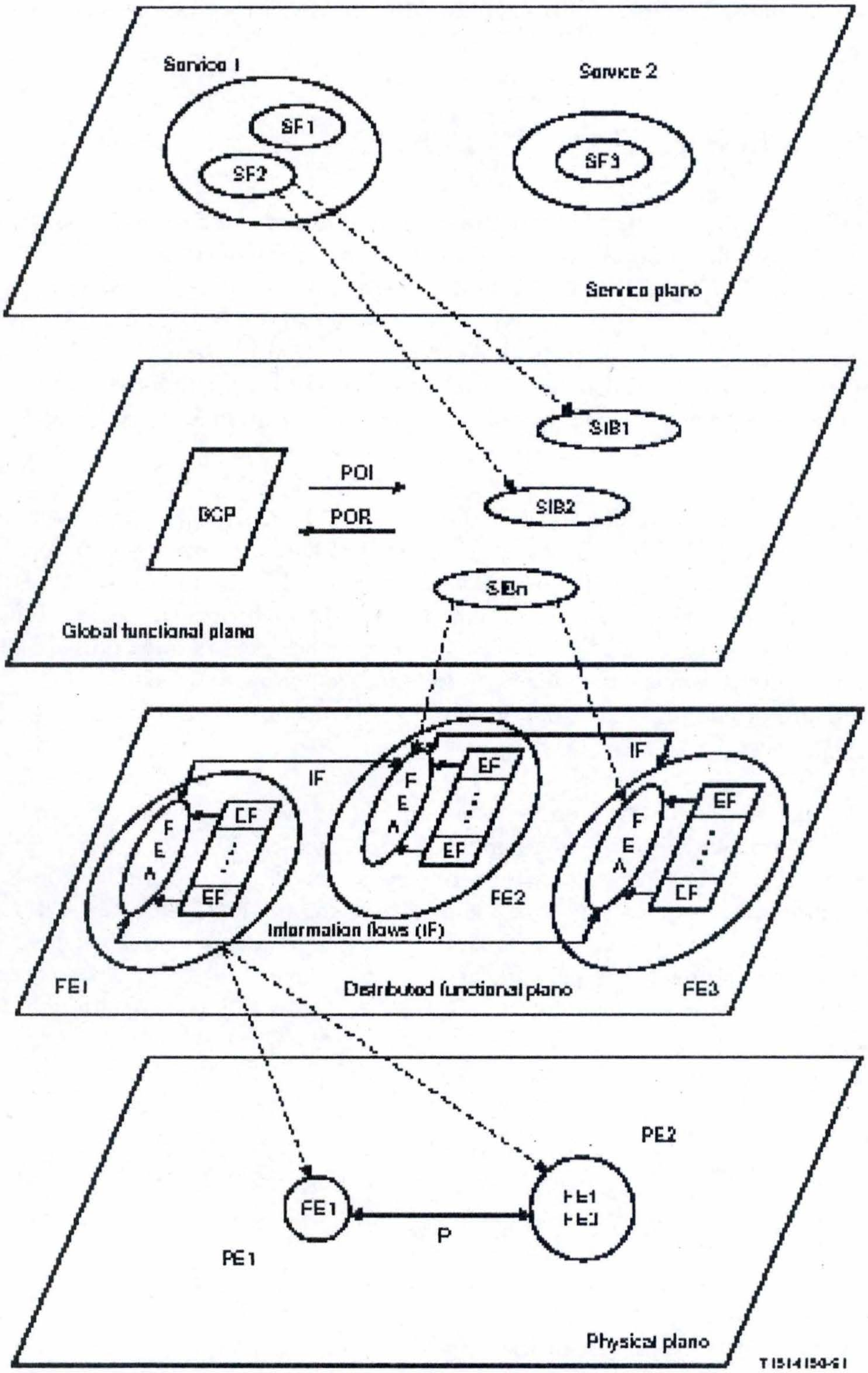
separated from the network has never been fully achieved. By the end of this chapter, you may see why.

2.4 The IN architecture

You may recall that the AIN concept is recognised as an industry standard in North America and that the International Telecommunication Union (ITU) has developed an equivalent version of AIN called Capability Set 1 (CS1), which comes in evolutionary subsets called the Core INAP capabilities. At the time of writing, 3 Capability Set versions are released and work on the fourth is in progress. Each Capability Set corresponds to a description of the IN conceptual model as well as the protocols that are used between the different entities. In this section, we provide a very basic overview of what is behind the capability set notion and we dwell on the IN conceptual model.

The IN architecture has been standardized by the ITU-T in the Q.12xx Recommendations series. The reader may consult [Q1200] in order to see how this standardization is structured. Another term is used worldwide to define the IN architecture, i.e. the IN platform. A platform is a combination of software and hardware components. The IN platform supports a huge number of features that are common to all IN services (such as billing, accounting, authentication, etc.). Thus, when a new service is installed, only the parts that are unique or specific to the service need to be developed and placed on the existing platform. This drastically reduces the time required to implement new services.

The IN conceptual model defines the standard architecture of an intelligent network. This architecture, in accordance with Open Distributed Processing (ODP, see [X950]) principles, is viewed in terms of planes. There are four of them: service plane, global function plane, distributed functional plane and physical plane. According to [Q1201], "the IN Conceptual Model should not be considered in itself an architecture. It is a framework for the design and description of the IN architecture [...]". Each plane represents a different abstract view of the capabilities provided by an IN-structured network. The IN Conceptual Model (INCM) is schematised in Figure 2.3.



BCP	Basic call process
EF	Elementary function
FE	Functional entity
FEA	Functional entity action
IF	Information flow
P	Protocol
PE	Physical entity
POI	Point of initiation
POR	Point of return
SF	Service feature
SIB	Service independent building block
- - ▸	Pointer

Figure 2.3 – IN Conceptual Model (see [Q1201, figure 20])

2.4.1 Service plane

The service plane is described in [Q1202]. It represents the service-oriented view. This view contains no information regarding the implementation of the services in the network. IN-supported services can be described to the end user or subscriber by means of a set of generic blocks called Service Features (SF). Thus a service is composed of one or more Service Features, which are the "lowest level" of services. Management services are also contained in the service plane.

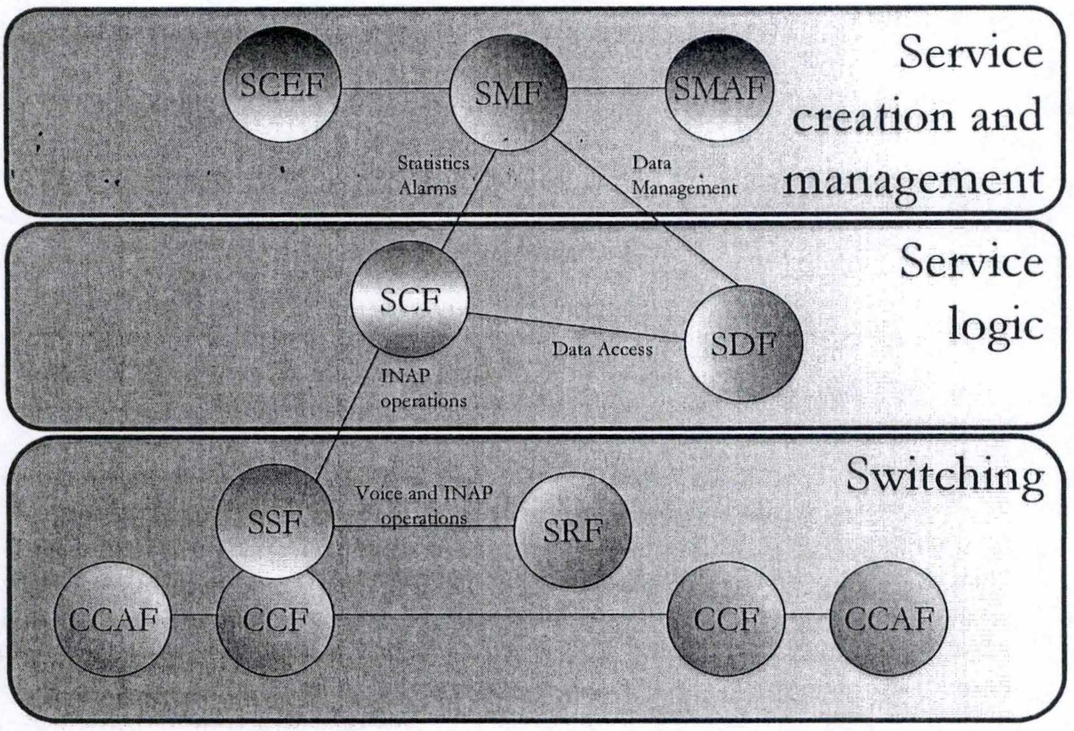
2.4.2 Global functional plane

The global functional plane, which is defined in [Q1203], models an IN-structured network as a single entity. Services and service features are redefined in terms of functions called Service Independent building Blocks (SIBs). The term "Independent" stresses the fact that SIBs are neither service nor SF-specific. In this plane, two specialised SIBs must come to the fore. The first, the Global Service Logic (GSL) allows to SIBs to be chained together with the aim to achieve a service (or more specifically, a service feature). The second, the Basic Call Process (BCP) contains the functionalities needed to manage a call. This management uses two synchronization point types: Points Of Initiation (POI) specify at which time of the call one may call the GSL. Points Of Return (POR) give the points in the call where the GSL can reactivate the service process. SIB is a very important concept in terms of reusability and we will come back to this notion when we have defined the "Internet version" of such services.

2.4.3 Distributed functional plane: a functional view of the IN architecture

The concepts defined here form part of the ITU-T Recommendation Q.1204 (see [Q1204]). Definitions are based on the IN Glossary document from Alcatel (see [ALCA00d]). The purpose of this plane is to identify groups of functionalities, referred to as functional entities, and the information that flows between these entities. Each functional entity may perform a variety of Functional Entity Actions (FEAs). Any given action may be performed within different functional entities. Within each functional entity, various actions may be performed by one or more Elementary Functions (EFs). The functions defined in the SIBs are performed in the functional entities by a sequence of FEAs. Some of these actions result in Information Flows (IFs) between functional entities.

Figure 2.4 shows the main functions of an IN platform architecture. This platform is vendor independent. It is very important to distinguish the difference between functions and nodes, i.e. between the distributed functional plane and the physical plane.



CCAF: Call Control Agent Function

CCF: Call Control Function

INAP : Intelligent Network Application Part

SCEF : Service Creation Environment Function

SCF: Service Control Function

SDF: Service Data Function

SMAF : Service Access Management Function

SMF: Service Management Function

SRF: Special Ressources Function

SSF: Service Switching Function

Figure 2.4 – Main functions of an IN architecture

The Call Control Agent Function (CCAF) provides network access functions for users, interacting with the Call Control Function (CCF). The CCAF is the interface between users and network call control functions. The CCF is the call control function in the network that provides call/connection and control. It establishes, manipulates and releases call/connection as "requested" by the CCAF. It also provides trigger mechanisms to access IN functionalities (e.g. passes events to the SSF).

The Service Switching Function (SSF) associated with the CCF provides the set of functions required for interaction between the CCF and a Service Control Function (SCF). It extends the logic of the CCF to include recognition of service control triggers and to interact with the SCF. The SSF manages the signalling between the CCF and the SCF as well. In other words, the major functions of the SSF are to invoke call and non-call related services on the IN platform and give the functions needed to allow the CCF to interact with the SCF.

The Service Logic Execution Environment (SLEE) runs call related and non-call related IN services, interacting with the SSF. The SLEE contains Service Execution Processes (SEPs) and the database for a given service. A SEP is an operational part of a service. It should be noted that the SLEE is not specified in Figure 2.4. The reason is that the SCF is a dedicated SLEE that commands call control functions in the processing of IN service requests. In the rest of this chapter, we will consider the SCF rather than the SLEE because the SCF is a far more extensively used term in the references of this dissertation.

The Service Control Function (SCF) commands call control functions in the processing of IN provided and/or custom service requests. The SCF contains the logic and processing capability required to handle IN provided service attempts. To provide these functions, the SSF may interact with other functional entities to access additional logic or to obtain information required to process a call/service logic instance. Typically, the SCF must contact the Service Data Function to translate a free-phone number (800) to its real phone number.

The Service Data Function (SDF) is a particular service aimed at being used as a data server (and even object server) by other services. It contains customer and network data for real time access by the SCF in the execution of an IN provided service. It interfaces with SCFs as required (and with other SDFs, if necessary).

The Specialised Resource Function (SRF) provides the specialized resources required for the execution of IN provided services. Examples of resources are announcements, Dual Tone Multi Frequency (DTMF) sending and receiving, speech recognition, etc. The SRF interfaces and interacts with the SSF and the SCF. These resources are clearly accessed by users (for example a voice invites the user to make a choice and the DTMF received determines what feature of the service the user has requested based on the key the user has pressed).

The Service Management Function (SMF) is involved with activities for service deployment, service provisioning, control, monitoring and billing. The SMF manages, updates and/or administers service related information in SRF, SSF and SCF. Furthermore, the SCF coordinates different SCF and SDF instances. Billing, statistics and alarms information are received from the SCF (in what is called a ticket) and made available to the authorized service provider through the SMAF.

The Service Management Access Function (SMAF) provides an interface between the service supplier and the SMF. This interface allows them to manage their services.

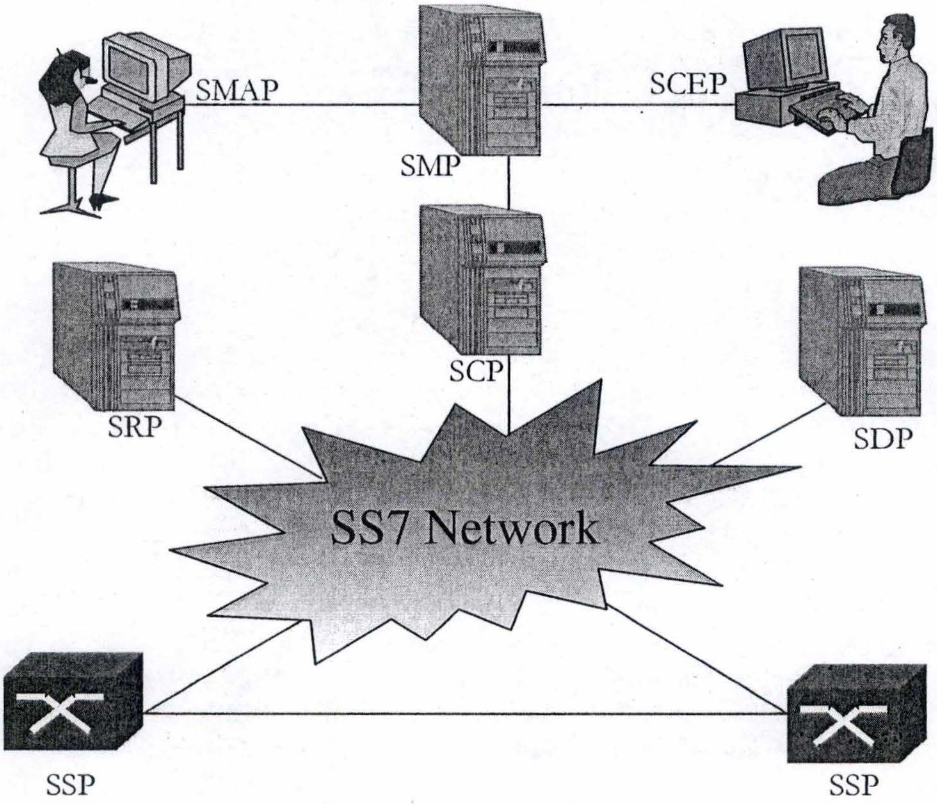
The Service Creation Environment Function (SCEF) is a set of functions that support the service creation process. It allows the defining, developing, testing and input to SMF services to be provided in an intelligent network. Output of this function would include service logic, service management logic, service data template and service trigger information.

2.4.4 Physical plane: a component view of the IN architecture

The physical plane models the physical aspects of an IN-structured network. The model identifies the different physical entities (PE) and protocols (P) that may exist in real IN-structured networks. It also indicates which functional entities are implemented in which physical entities. Even if this physical aspect can be viewed in different ways, the network must possess given properties in order to be consistent with the IN Conceptual Model. Some of these properties are (see [Q1205] for more details on the physical plane):

- One functional entity corresponds to one physical entity
- One functional entity cannot be split between two physical entities
- Physical entities can be grouped to form a physical architecture

Figure 2.5 shows a possible implementation of an IN-network. The implementation we propose here is the one used when powerful performances must be provided. Roughly, to each functionality corresponds a network component dedicated to the implementation of this functionality.



SCEP : Service Creation Environment Point

SCP: Service Control Point

SDP: Service Data Point

SMAP : Service Access Management Point

SMP: Service Management Point

SSP: Service Switching Point

SRP: Specialised Resource Point

Figure 2.5 – Main IN network components

The Service Switching Point performs switching functionality and allows access to the set of IN capabilities. The SSP hosts the SSF and CCF functionalities. Furthermore, it provides users with access to the network when located in the local exchange. In that case, the SSP also contains the CCAF functionality. The SSP can optionally contain an SRF or an SDF or both. The SSP contains enough capabilities to communicate with PEs containing an SCF. In short, the three main tasks of the SCP are trigger detection, command execution and event handling.

The Service Control Point (SCP) hosts the SCF and contains Service Execution Processes (SEPs) in order to provide the requested service. An SCP can optionally contain an SDF and can access data in an SDP either directly or through a signalling network (e.g. SS7).

The Specialised Resource Point (SRP) hosts the SRF. In older literature, the SRP is often called the Intelligent Peripheral (IP). The first term has been chosen to avoid confusion with the Internet Protocol (IP).

The interested reader may consult [Q1205] in order to have more details on the physical plane. It is important to stress that because an IN component does not match necessarily with an IN functionality, it is better to use the functional term (e.g. SMF) rather than the component one (e.g. SMP). Of course the component term must be used to define the component functionality as a whole (e.g. the SSP term regroups the SSF and CCF functionality at least)

2.4.5 The Basic Call State Model

The ability to open the basic switching process to external influence is a prime necessity to allow service-independence to take place. This is done by defining a call model, called the Basic Call State Model (BCSM). This model describes the CCF activities required to establish and maintain communication paths for users. Furthermore, the BCSM identifies points in basic call and connection processing when IN service logic instances are permitted to interact with basic call and connection control capabilities. The components that have been identified to describe the BCSM are Points In Call (PIC), Detection points (DP), transitions and events. "PICs identify CCF activities required to complete one or more basic call/connection states of interest to service logic instances. DPs indicate points in basic call and connection processing at which transfer of control can occur. Transitions indicate the normal flow of basic call/connection processing from one PIC to another. Events cause transitions into and out of PICs"[1]. Such a call model is defined for every capability set, but the interested reader may consult [Q1204, pp. 16-25] in order to have a general view of this notion.

According to [Q1204] some functionality must invoke capabilities provided by other functionalities in order to support the execution of a specific IN-supported service feature. For example, the SCF must invoke capabilities provided by the SSF. Moreover, "in order for one functional entity (termed the client functional entity) to invoke the capabilities provided by another functional entity (termed the server functional entity), a relationship must be established between the two functional entities concerned". This mechanism is provided by the Capability Set (CS), which leads to the Intelligent Network Application Part (INAP) concept. The INAP defined for CS-1 in [Q1218] is the protocol used between the following functionalities: SSF, SRF, SCF and SDF.

Unfortunately, many versions of INAP exist throughout the world. In fact, there is far more than one version per capability set. Vendors have their own IN implementations, many of which have proved to be incompatible. This is accentuated by the fact that they create different equipment. In short, each vendor has their own view of an IN and most of the time, these views are incompatible with each other.

2.5 Examples

First of all, we provide here a generic scenario for an IN call. Then practical examples are given: free phone (800) and wake-up services. The reader must absolutely keep in mind that such services are implementation-dependent. We try, however, to be as generic as possible.

[1] [Q1204, p. 16].

2.5.1 Generic scenario for an IN call

A user who wants to access an IN service must first access the telephone network via a switch in the local exchange, for example. This switch may contain software enabling it to be configured as an SSP or may, after identifying the call as an IN related call, direct it to the appropriate SSP. The SSP can recognize a call as an IN-related call because of a Service Access Code (SAC). A SAC contains the prefix code of the desired service. The SSP uses the SAC information and criteria in its trigger tables to analyse the call. In the case that the switch in the local exchange is "SSP-compliant", the triggering of the SCP can also start from the switch itself.

Throughout this chapter we have seen that modern IN tends to take the intelligence, i.e. the service logic, out of the switch. Once an IN call has been detected by the SSP, the handling of this call is shared between the SCP, which controls the overall operation, and the SSP, which forwards the information to and from the SCP. Remember that an SSP contains two functionalities needed to relay calls in the IN: the CCF and the SSF.

After receiving the call to a service, the SSP analyses it and determines where the SCP for that service is physically located in the IN network. Then the SSP gathers the information needed to proceed with the call (for example, the caller's identity). Afterwards the SSP requests the SCP responsible for the service to take charge, and forwards the needed information, which is conveyed using the INAP protocol.

When the SCP receives the request to intervene in a service call, it selects the service script based on the input message from the SSP. Remember that the call handling is ensured by service scripts and SIBs. The software in the SCP analyses the script of the called service, launches the elementary actions indicated in the script, monitors their execution and remotely controls the SSP. The SCP can, for example, instruct the SSP to play a recorded announcement through a connection with the SRP or to establish a connection between a network input and a network output.

Besides interacting with the SSP, the SCP records all service related data in its database. This data, called the ticket, is automatically updated into the SMP database and can be used for statistics and, optionally, charging.

2.5.2 Free phone (800)

The free phone service is a well known one. Basically, the purpose of this service is to allow users to establish a call for free. Such calls are charged to the called party. 800 numbers, such as 0800/123456, are not true phone numbers. In fact, the IN service logic translates it to a real number, such as 03/111223344. Towards the evolution of the IN architecture, the free phone service has been deeply improved. Modern service can translate 800 numbers depending on various factors such as time of day or day of the week. Figure 2.6 shows how the IN architecture can handle a free phone call. SSPs form part of the IN platform and thus the cloud named "IN platform" brings together the remainder of the platform, such as SCPs, SRPs, SMPs, etc.

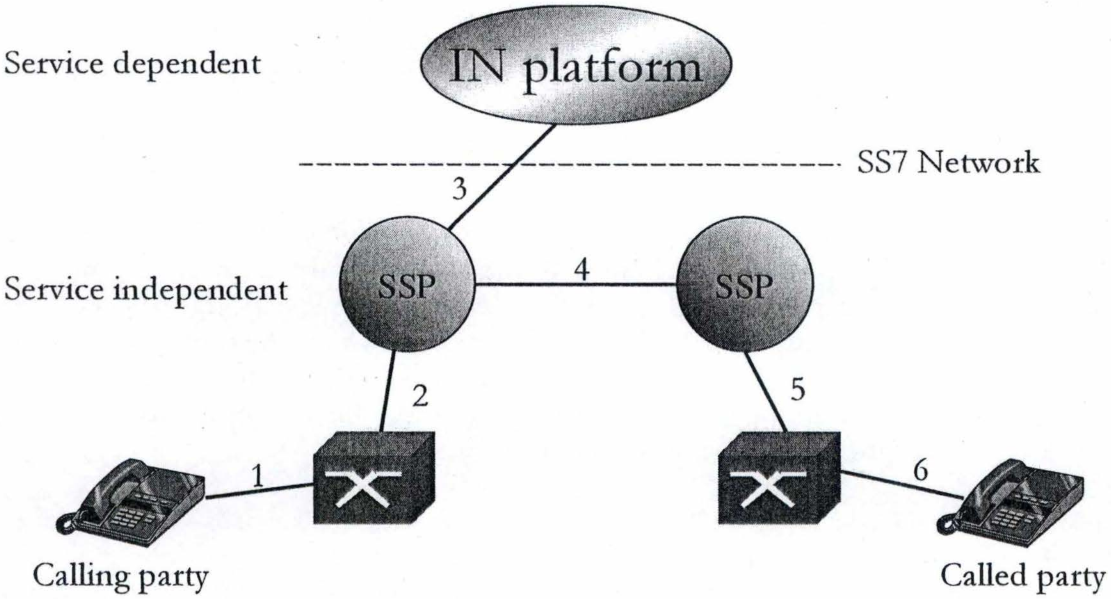


Figure 2.6 – Freephone example

Assume that the calling party wishes to call a freephone number (e.g. 0800/123456). To do so, the CPE of the calling party accesses the local exchange through the local loop (1). The local switch is not configured as an SSP and does not know this number because it is not an existing directory number. Thus, in 2, it forwards the call to a higher level, a tool exchange, for example, which is configured as an SSP. This switch is still unable to handle the call but it contains a specific trigger for this kind of number. The Intelligent Network is triggered to execute the freephone-software and thus handle the call. In 3, the SSP forwards the needed information to the IN platform through the SS7 network and by using the INAP protocol. The platform translates the number into the directory number of the called party, e.g. 03/111223344. This information is transmitted back to the SSP in the same way. Now the tool exchange is able to process the call and contact, in 4, the tool exchange of area 03. In 5, the switch of area 03 contacts the local switch to which the called party is connected. In 6, the calling party and the called party are connected.

We have not described what happens when the freephone software translates the 800 number. And the power of IN is simply that the SSP does not care! Ways and factors that involve the translation have potentially no limits. It should be noted that this example does not represent what happens between the SCP and the SMP. Even if the call is free for the calling party, someone must pay for that call. As we have seen previously, the complete transaction is summarized in a ticket that is sent to the SMP. This ticket is then used to charge the called party for the duration of the call.

2.5.3 Wake-up service

The purpose of the wake-up service is to replace (or be used as) a radio alarm. A lot of businessmen use it when they are on business trips, for example. While the majority of IN services are triggered in the SSP, the wake-up service is a typical example showing a service being activated at the SCP-side. In fact, the user must first register with the wake-up service. Typically, (s)he provides the date and time where (s)he wants to be woken up. Then, at the appointed date, the SCP instructs the SSP to establish a call between the user's CPE and an SRP playing a recorded voice related to this service.

2.6 Conclusion

Intelligent networks are widely used in telecommunication networks all over the world. The aim of the IN architecture is to facilitate the creation of services by providing a device-independent network in which services are separated from the network. Unfortunately, this has not been fully achieved for two main reasons. First of all, the fact that vendors have their own views of what an IN is, many of which have proved to be incompatible. Then there is the incredible number of INAP versions that exist throughout the world. However, intelligent networks have made headway and provide powerful services to customers.

Unfortunately, the classic CPE, i.e. a simple telephone with 12 keypads (0-9,#,*) is rapidly proving to be restrictive in many ways. One of the biggest limitations is the poor ability for customers to customize their services. In fact, vendors have created everything one may want to do with a simple telephone. But this limitation is not the sole cause for IN to change. As we shall see in chapter 3, the network landscape is changing. Initially, providing voice services on packet networks has the consequence that IN must interact with packet networks in order to provide these new customers with the same set of services that a PSTN user can utilize. Thus IN is a major actor for network convergence. In the next chapters, we discuss how IN can handle and facilitate this convergence.

Chapter 3

Interworking IP and the PSTN

The users of Internet telephony must have the possibility to reach and be reached by the users in the PSTN. After all, this assessment is the foundation of the notion of "network convergence". Actually, this interworking is the first phase towards the Next Generation Networks. In this chapter, we cover the architecture needed to interwork IP and the PSTN. First we provide the requirements of voice and data traffics, which are fundamentally different. These differences involve the reason why these networks must evolve in order to interwork. Then we study how the PSTN and IP-based networks can actually be modified in order to provide a good architecture enabling the creation of convergence services. This architecture cannot exist without the gateway concept that we discuss intensively afterwards. Furthermore, Internet telephony cannot afford to forgive the existing services in the PSTN. We end this chapter by showing how existing services can be supported and provided.

3.1 Introduction

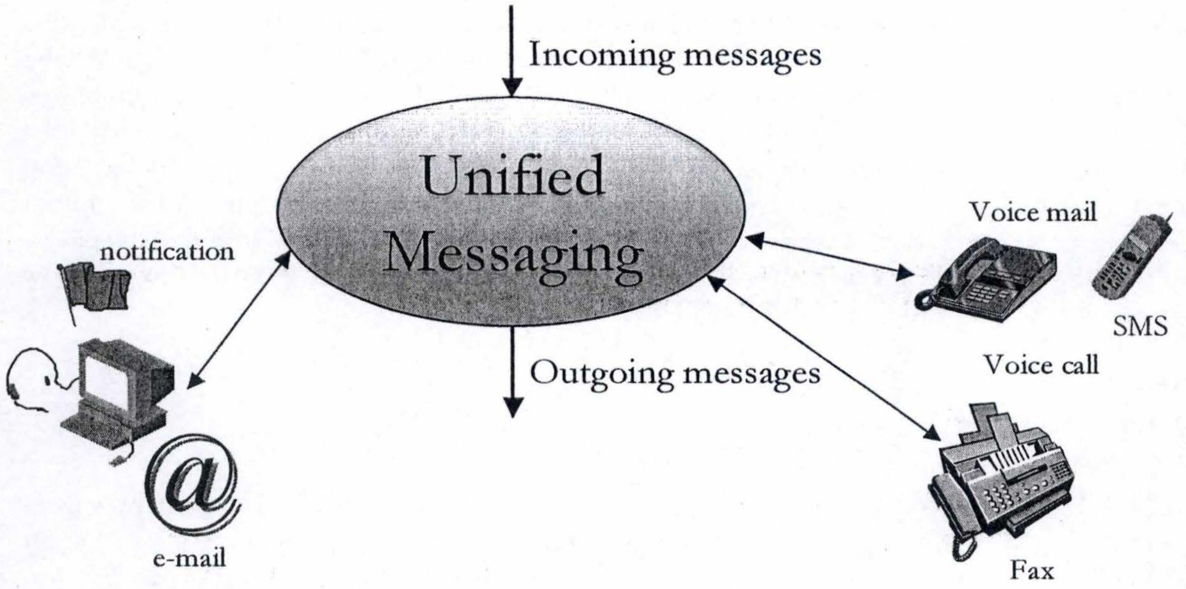
In this introduction, we analyse the following question: "Why the PSTN and the Internet must evolve to support Internet telephony?". Before answering this question, it is interesting to point out some properties and requirements of voice and data traffics on a packet-based network.

Real-time traffic	Data traffic
End-to-end delay must be low	End-to-end delay is not a critical issue
Occasional loss of packets has little or no effect on voice quality	Loss of packets has always an impact
Packets retransmission must be avoided	Packets retransmission is necessary
Jitter must be low	Jitter is not a critical issue

Table 3.1 – Real-time traffic and data traffic requirements/properties

This table is not exhaustive but it gives an answer to our initial question. This answer is based on the fact that, whenever an architecture is designed, it is always done with well-defined applications in mind. For the Internet architecture, the purpose was data transfer, i.e. file transfer, virtual terminal for remote access, e-mail and of course web-browsing. Although web browsing was not in the initial internet-related applications, it is the most widely used today with e-mail and they have driven many architectural decisions. Today's Internet provides a good architecture to support these services and has been engineered with the bursty pattern of data traffic in mind. However it is not designed, as shown in Table 3.1 above, to support efficiently real-time traffic (e.g. voice traffic).

Even if the PSTN has already a packet mode (e.g. ISDN), it has to be modified as well due to the importance of dialup access to the Internet, for instance. In this dissertation we use the "service" term to define the IN services that exist in the PSTN (and which are limited by its architecture!). We use the "advanced service" term to define the new generation of services that can be created with a more powerful architecture like the Internet. An example of such advanced service is the Unified Messaging service. Its purpose is to eliminate the boundaries across the different forms of messaging. In other words the purpose of this service is to give the ability to create, send and retrieve any type of message with any terminal anytime, anywhere (Figure 3.1). In the next chapters, we study how we can develop advanced services in this new environment.



SMS: Short Message Service

Figure 3.1 – Unified Messaging

We conclude this introduction by giving some useful vocabulary in order to understand the remainder of this chapter. We first recall the reader to what "hybrid" means in the scope of this dissertation. A hybrid network is a network involving two kinds of networks. The most prevalent example here is the hybrid network that involves IP networks and the PSTN. In such a network, Internet telephony can be classified into four types based on the endpoint's devices involved in the call. When both devices are traditional telephones, we use the term "phone-to-phone". When the originating device is an Internet phone and the terminating one is a traditional telephone, we use the term "PC-to-phone". The reverse situation is called "phone-to-PC". Finally when both devices are Internet phones, we use the term "PC-to-PC". These concepts are extensively defined in [FGL00, pp. 198-204].

3.2 Modifying the PSTN

The growth of the Internet has had major consequences on the PSTN. The PSTN was not initially designed to carry the traffic coming from the Internet and thus it must evolve for that very reason. Secondly, the bandwidth of 56kbps provided by a standard PCM-modem is insufficient to support the new generation of services that can be settled in the architecture we discuss here.

An important problem facing the PSTN today is the data traffic that it carries to and from IP networks. The PSTN was originally designed to provide short calls with a good quality that is to say no distortion, no echo, etc. Unfortunately, Internet access calls last much longer and thus tie up the resources of local and tandem switches. As a consequence such calls increase the number of uncompleted calls in the PSTN[1]. Furthermore, data is bursty while voice occupies the whole bandwidth permanently. Thus the 64kbps bandwidth allocated by each call in the PSTN is not used efficiently. The solution to these problems is to identify Internet traffic and offload it to a packet network as soon as possible. This conflict between data and voice in the PSTN has created a classical example of PSTN-Internet integration by necessity.

End-users who want to access the Internet must contact their ISP by calling an Internet access number. The solution is to identify these numbers and to use IN to create a special trigger in the switches. When a user calls one of these numbers, the trigger activates an IN service that instructs the switch to route the traffic of this user to a packet network. [FGL00, pp. 51-53] provides a good explanation of this problem and possible solutions.

The great majority of users access the Internet with a 56kbps standard modem. Even if the voice can be compressed thanks to codecs, this bandwidth is clearly insufficient to support voice calls with an acceptable quality. Furthermore, the advanced services will most probably require higher bandwidths. To solve this problem, new types of access to the Internet have been created. The most important one is the xDSL technology. The interested reader may consult [FGL00, pp. 43-50].

They are another reasons why the PSTN must be modified, more economical ones. For instance, some companies wish to route their international voice traffic to data networks under leased lines in order to spare money (this is defined by the "phone-to-phone" concept and is more known as the Virtual Private Network (VPN) concept). Moreover the data contained in the PSTN such as 800-database must not be cloned for the Internet telephony environment. It means that a way must be found to contact such a database in order to provide 800-services and others on the Internet as well.

3.3 Modifying IP networks

This section studies important protocols that can enable IP-networks to provide acceptable voice quality. Remember Table 3.1. First of all, there is no Quality of service (QoS) concept in the majority of IP networks. The purpose of the introduction of QoS in IP networks is to shorten the end-to-end delay as well as the jitter that can occur during a call. To achieve this, the QoS may be capable of reserving the needed bandwidth in the network to avoid packet loss as well. Indeed, packets are generally lost because the outgoing lines of a router are congested due to the traffic. Thus if the bandwidth is reserved in any way along the path, it decreases the probability of losing a packet. The second protocol is related to label switching that can improve, together with QoS, end-to-end delays.

3.3.1 Quality of Service

Quality of Service (QoS) is one of the most important issues for providing telephony features on an IP network. QoS addresses both objective and subjective aspects, that is to say performance

[1] The PSTN can block calls to a very busy switch and send special tones to the customer to notify her/him.

of a link and voice quality. Please note that different applications have different perceptions of what QoS is. For instance, telemedicine QoS is not the same as voice QoS. Indeed, "the transmission of a brain X-ray taken in a remote laboratory can be delayed for a few minutes but the data cannot be compromised. Any wrong details can be interpreted as a tumor or leave it undetected" ([FGL00, p. 60]). QoS can be approached by two ways: the first is to have a QoS on a per-flow basis; the second is to have a QoS on a per-packet basis. The first way is taken by the Integrated Services (intserv) with Resource Reservation Protocol (RSVP). The other way is taken by Differentiated Services (diffserv).

In the integrated services approach, applications must express their QoS requirements through Resource Reservation Protocol (RSVP). The purpose of RSVP is to reserve some resources in the network while establishing a flow. An RSVP state is associated with each flow; these states are stored in routers in soft-state mode[1]. The advantage of using soft-state mode is the ability to deal with route changes[2]. It is important to point out that integrated services are almost never implemented in practice. The most widely accepted reason is that the mechanisms associated with reservations are far too complex to provide a good general performance. It results that short-lived flows are at a disadvantage with this approach even if the model was not created with the aim of solving the problem of short-lived flows connections.

In differentiated services, the purpose is to provide a QoS on a per-packet basis, i.e. ordering traffic aggregates. The packets are classified based on a certain target behaviour, a class of service. The service provider establishes with each customer a service level agreement (SLA). For example, a SLA specifies how much traffic a user may send within any class of services. A particular class of service of a packet is coded in its IP header: this is the differentiated services code point (DSCP). The border router does this coding and backbone routers rely on this information to provide the different types of service supported within the differentiated services domain (DS domain). This approach has the advantage, unlike the integrated services approach, to free the backbone routers with the management of data flows. Complexity is located at the border of the DS domain and treated by specialised network equipments. QoS is a rather hot topic and the reader may find a lot of literature on this subject. (S)he may also consult the bibliography of this chapter to find additional references.

3.3.2 Label switching

An important requirement of voice traffic is the network's ability to send packets in sequence and with an acceptable end-to-end delay. Now in the current architecture, packets are sent on a routing table basis (so packets can follow different routes based on the network status). A first approach is to use an ATM (Asynchronous Transfer Mode) or frame relay network under the IP network to provide circuit-switching facilities. However, this solution has too many drawbacks to be used in practice: first, it implies the management of two networks, i.e. the IP network and the circuit-switched underlying network. Then IP is too widely deployed in the world to be changed this way.

Another method is Multi Protocol Label Switching (MPLS). The MPLS standard has been developed at the IETF (see [MPLSWG]). At first, it was developed to allow the integration of ATM with IP networks. But the drawbacks given above have driven the IETF to adapt MPLS

[1] A timer is associated with each flow. Thus in soft-state mode, routers keep the information till the timer expired. End systems associated with the flow must send periodically keep alive messages to reset the timer.

[2] When the route changes, the end systems stop sending keep alive messages to the old route and the routers delete the old state once the timer has expired.

with the aim of integrating all data link layers with current network layers. The term multiprotocol has been chosen to notify this objective. In a nutshell, MPLS is a connection-oriented forwarding scheme using Forwarding Equivalence Class (FEC). A FEC is a set of packets associated with a path starting at a given router. With MPLS forwarding, each packet is assigned to a FEC only when it enters the network. Then the backbone routers rely only on the FEC to forward the packet inside the network[1]. IP headers are completely ignored. It allows the use of a virtual circuit inside the IP network. Moreover, the semantic of the FEC (represented as a label) can be changed to reflect the prioritisation established by diffserv or intserv. In this case, the label is constructed by a combination of a FEC and the prioritisation in question. New ways of routing in IP networks is an important issue today: readers can consult [DARE00] to have more details on IP over ATM networks and the MPLS standard as well.

3.3.3 Basic Quality of Service efforts?

In other words, are these standards, that is QoS and label switching, enough to provide voice-delivery in IP networks? Of course not, and for many reasons. First of all, we have forgotten here an important standard that is widely used in IP-networks: the Real-time Transport Protocol (RTP see [RFC1889]). RTP is designed to support real-time traffic and runs on top of UDP. It provides a wide range of services and is extensively used by Internet telephony. A good description of RTP in the IP call processing environment is given in [BLAC00, pp. 100-118]. Then, one should not forget that these solutions are not suitable in all cases. We have provided them only to present the current trend concerning QoS researches.

After the protocols that enable IP-networks to provide voice features, they are a number of basic differences between IP networks and the PSTN that cause issues. First, while the addressing scheme in the PSTN is based on point codes for network nodes (see section A4.5.5) and E.164 addresses (see [E164]) for endpoints, other addressing schemes are used in the Internet: IP address, Domain Name System (DNS), Uniform Resource Locator (URL), etc. A second important difference is voice representation: G.711 (see [G711]) in the PSTN while many other representations are possible in the Internet, especially compressed voice representations like G.729 (see [G729]). Thirdly, an important one, signalling protocols are different: mainly SS7 in the PSTN while H.323 and Session Initiation Protocol (SIP) are the most prevalent solutions for the Internet. These signalling protocols for communication setup and teardown over IP networks are discussed in appendix 5. If you are unfamiliar with SIP, we urge you to read this appendix. All the differences that have been covered here imply the necessity of an entity, called a gateway that compensates, together with IN for instance, for these differences. This entity is the subject of the next section.

3.4 IP telephony gateway

The gateway is the glue that links the PSTN and IP networks together. Several architectures for IP/PSTN integration have been created and we give here only a short description of them. Then we study in more details the current research in this field and we give the issues and solutions related to them. The discussion below is strongly driven by our first objective: providing IN services on a hybrid network.

[1] It implies that FEC have, in general, only a local significance.

3.4.1 A short history

The convergence of the PSTN and IP networks promises exciting opportunities for local and long-distance calls. An important step in the accomplishment of this objective is the extension of IN capabilities to and from IP networks. As you may recall, IN services are controlled by the INAP protocol through the SS7 network. Logically, manufacturers plan to build IP telephony switches that support SS7. The first one is a small switch with an SS7 stack configured as an SSP. The architecture is rather simple. Each IP telephony switch is connected directly to the SS7 network, enabling the support of IN services. Unfortunately, this solution cannot survive at mid-term. A straightforward reason is that each switch requires an SS7 point code[1].

The next step for these manufacturers is simply to build bigger IP telephony switches. This resolves some problems but it has also the drawback of being less reliable. Indeed, as the switch gets bigger, more traffic flows through a single switch and then a crash has more consequences. Furthermore, this architecture ties up the local resources, as users place Internet telephone calls from various switches. Moreover we have seen in the second section of this chapter that solutions are created in order to offload the Internet traffic to a packet network as soon as possible. Thus improvement in the gateway's architecture must be found. The gateway decomposition is the answer to these shortcomings.

3.4.2 Gateway decomposition

The "modern" gateway notion is rather inaccurate. [RFC2719] gives a generic model of the gateway which contorts its features into 3 functionalities, as shown in figure 3.2. We try here to describe these functionalities and we show how they can be actually contorted within devices. Then we discuss the problems enounced in 3.3.3. The gateway location is also an important issue and must be correctly defined in order to contact the PSTN user as efficiently as possible.

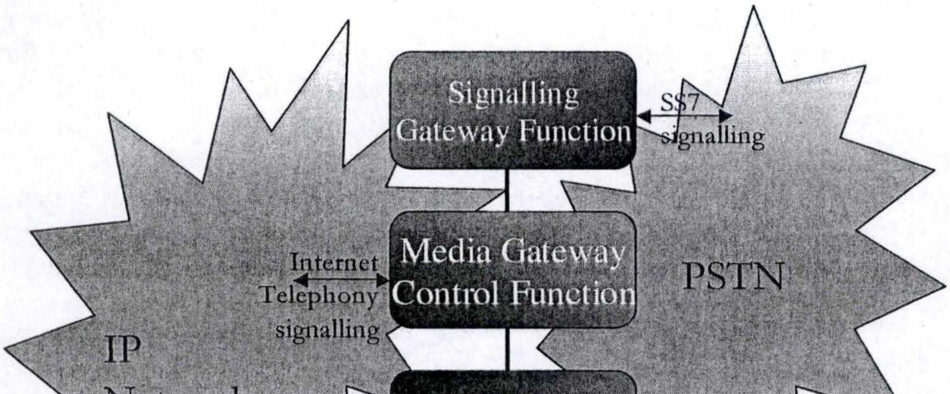

Signalling Gateway Function
SS7 signalling
Internet Telephony signalling
Media Gateway Control Function
PSTN
IP Networks
RTP streams
Media Gateway Function
Media streams

Figure 3.2 – Gateway functionalities

[1] The reader may recall that an SS7 point code identifies an SP in a unique way inside the SS7 network (see A4.4). Point codes are strongly limited inside an SS7 network (2^{16} possibilities).

As you can see, there are three main components in this architecture: the Media Gateway (MG) function, the Media Gateway Control (MGC) function and the Signalling Gateway (SG) function. This approach is clearly different from the traditional circuit-switched one. In the IP world, logical functions are unbundled and distributed as shown in the figure above.

The MG function terminates physically PSTN circuits and connections to IP routers from which they are connected. The main purpose of the MG function is to convert bit streams from one network into bit streams particular to the other network. This transformation occurs at two levels. First, at the transmission level, it involves on one hand multiplexing on the PSTN network and on the other hand demultiplexing in the IP network. As we have seen in chapter 2, voice channels are multiplexed based on the Time Division Multiplexing (TDM) scheme into frames. In IP networks, voice channels are packetized into RTP payloads[1]. Secondly, at the application level, voice-encoding schemes are different. While G.711 is mainly used in the PSTN, others voice codecs are used, mainly for bandwidth reduction purpose. This different encoding scheme has two consequences. First, voice quality is reduced (depending on the encoding scheme). Secondly, the computation needed to encode voice traffic causes delays. Thus, the MG function may support the use of the QoS facilities of the IP network. This function may also provide support for Internet offloading or advanced features such as playing announcements, collecting digits, statistics, etc. These advanced features may be off-loaded to a dedicated device.

The SG function is responsible for the signalling operations in the networks. For example, it may translate an ISUP message into an H.323 one. It has also the responsibility to translate the analog tones coming from the PSTN to binary numbers for transport over a data network and vice versa. It communicates with the MGC function using IP and with the PSTN using SS7. The roles of this function depend of the model used. In the trunking gateway model (see below), the SG function is only used to tunnel the signalling to the MGC which is responsible for the conversion.

The MGC function controls the whole system: it monitors the resources and controls all the connections. It is also responsible for authentication and network security. It originates and terminates all the relevant signalling. Most of the times, it also translates the signalling that comes from and to the SG function. Indeed, the SG and MGC functions are very often co-located. The MGC function is more known as the softswitch we discuss in more details later in this chapter.

3.4.3 Gateway scalability and distribution

The scalability of such systems is an important issue. These 3 functionalities can be combined in various ways. The following terminology is used: the Media Gateway Unit (MGU) is a physical entity that contains the MG function. The same goes for the Signalling Gateway Unit (SGU) and the Media Gateway Control Unit (MGCU) for the SG and the MGC functions respectively. Furthermore, different types of gateway in support of IP telephony exist. These types are distinguished by the functions they support. The first, trunking gateway, connects a tool office switch to an IP router. Typically, this gateway has an SS7 interface and manages a large number of connections (64kbps circuits for the PSTN-side, RTP streams for the IP-side). The trunking gateway is used to replace long-distance PSTN trunks by IP-networks, i.e. to make phone-to-phone calls. The access gateway connects telephones to an IP router through an access interface (e.g. an UNI). This gateway supports PC-to-phone (or phone-to-PC) and phone-to-phone calls.

[1] A payload of recommended size holds 20 ms of voice data. This choice is a compromise between end-to-end delay and header overhead.

The network access server connects a tool office switch to an IP router through an ISDN interface for example[1]. Finally, the residential gateway connects analog phones to an IP router. This gateway supports generally a small number of analog lines and is located close to the customers. Its purpose is to maximise the use of the IP network.

Gateway distributions (between the 3 functions we have seen so far) results in new interfaces between the components, such as an interface between the MGC and the MG function as well as an interface between the MGC and the SG function[2]. The first interface consists of a protocol that can be used by the media gateway controller to run the media gateway (in the scope of the MGC function's rights, i.e. call control, connection control and resource allocation). A protocol that has been developed jointly by the ITU-T and the IETF is the MEGACO protocol. The reader interested by this protocol may find a good survey of it in [BLAC00, pp. 185-225]. The other interface consists of a protocol that can transport SS7 signalling over an IP network. A well-known protocol is the SCTP protocol (see [RFC2960]) from the SIGnalling TRAnsport (SIGTRAN) working group at the IETF (see [SIGWG]). Figure 3.3 summarizes the concepts we have seen so far.

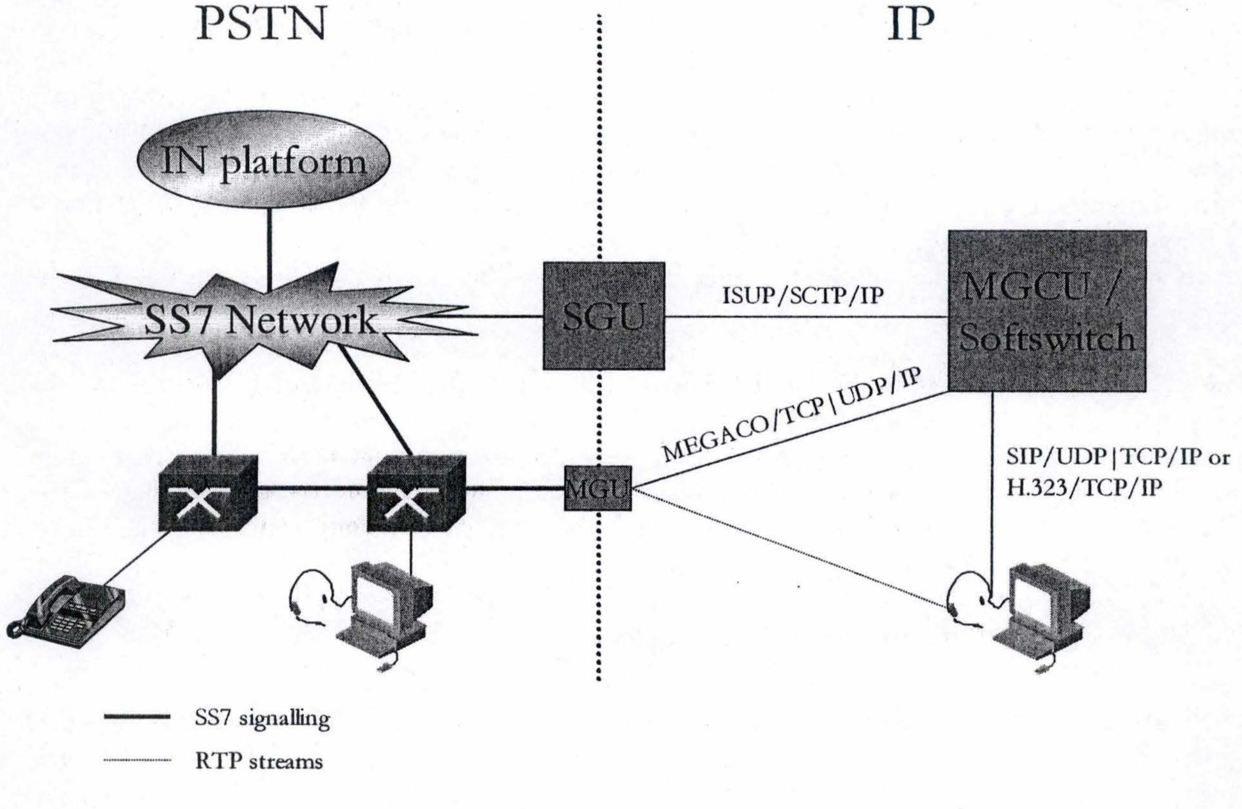


Figure 3.3 – A PSTN / IP interworking architecture

This figure does not show a multi-MGU model where a single MGCU can manage multiple MGUs. This model should be studied with care, since it is still evolving. For instance, even if the MGC is well defined, its interfacing with the PSTN is quite undefined. A number of working groups at the IETF, mainly the SIGTRAN working group, are working on these pieces of

[1] The reader may want to know the difference between the trunking gateway and the network access server. The purpose of the first is only to support phone-to-phone calls as the later supports more scenarios.

[2] This is not exhaustive. Many more protocols might exist, like interface between MGC components in a network.

standardisation. Please note also that the link between the MGCU/Softswitch and the IN platform is not schematised. We discuss this in section 3.5.

3.4.4 The softswitch in more details

The softswitch (see [SSCT]) can also be named a soft-SSP since it is configured as a traditional SSP. Thus it has the ability to launch queries to SCPs for instructions for further call processing, without any change to the SCPs at all. This entity can manage PC-to-PC calls on its own. When a call involves a user in the PSTN, the MG function is required (as well as the SG function, if it is not co-located with the softswitch).

According to [GRAP01], the traditional switches are "monolithic" in nature: they use proprietary protocols for communications between different subsystems. The advance in the NGN way is to use an open architecture with open interfaces between the pieces of this architecture. Of course, the softswitch must be open and programmable. On the one hand, it must allow the integration of numbers of protocols that exist in the IP world (SIP, H.323, MEGACO, etc.). By allowing the inclusion of a programmable stack for these protocols, it can be extended to lots of environments. On the other hand, the softswitch must be programmable for service providers and third-party feature developers. These concepts are schematised in Figure 3.4.

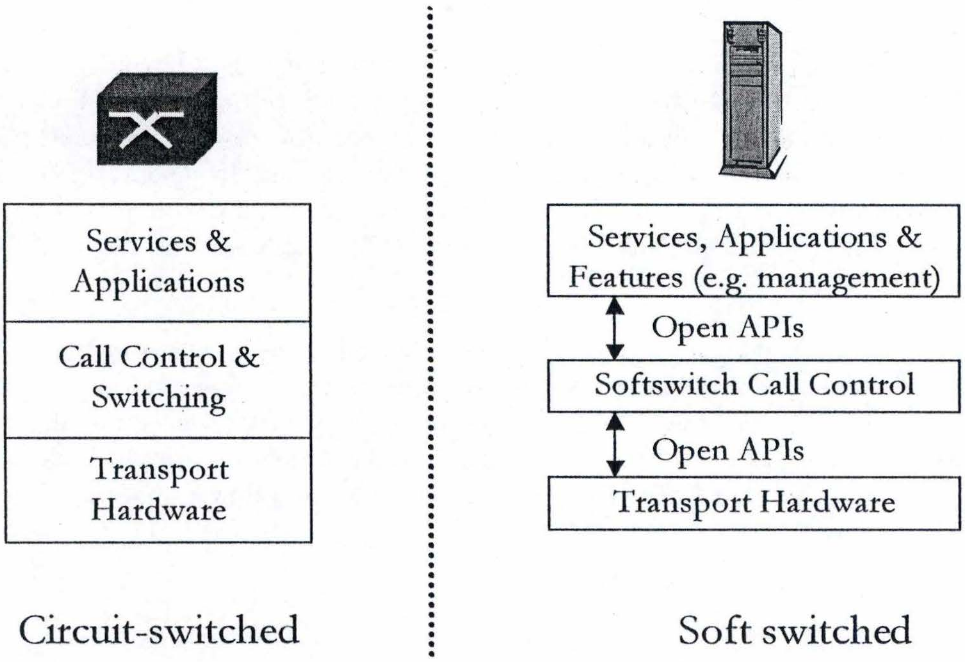


Figure 3.4 – Circuit-switched vs. Soft switched

As shown in the figure above, the next generation switching architecture is based on three logically separate layers. A layer can change independently of the others thanks to the APIs. The transport layer is responsible for the transmission of the bearer traffic: it may be IP or ATM, for instance. The softswitch resides in the second layer, the switching layer. This layer provides the control logic required for call processing and routing of the bearer traffic. The last layer, the service layer, provides the call logic and the databases for supporting telephony services. This layer can be compared to the SCPs we have studied previously.

Typical APIs between the transport layer and the switching layer are MEGACO, SIP and H.323. For what concerns the APIs between the switching layer and the service layer, we can find Parlay for instance which we discuss in chapter 5.

3.4.5 Gateway location

In the refined scope of voice transport, the location of the various gateways is important to find and to retrieve. Suppose that a pc-to-phone call must be established. It means that the called party resides somewhere in the PSTN. How can we reach this user? At great scale (and we suppose that it is the case), there are many gateways that could do the job, but only a few (or only one!) may be suitable for that call. By suitable we do not only mean here "that is as close to the called party as possible". Do not forget that they may have many other factors involved in such a choice: the price of the call, the current load of the given gateway or a local policy for example.

This problem is well known since a similar problem has occurred in the past with interdomain routing. The solution to interdomain routing is defined by the Border Gateway Protocol (BGP see [RFC1771]). The solutions we describe here are actually based on this protocol. The network is composed of domains. In each domain there is a special device called a location server. Its job is to learn about the gateways in its own domain as well as in other domains and to construct the database that the endpoints of this domain are going to use.

Discovery of gateways in other domains is not a straightforward task. Actually, there are many other factors other than the most basic one: "how can we reach the gateway and what range of PSTN users it can terminate?". Business agreement between domains, intradomain policy, etc. implies that a single database that all users can consult cannot exist. Instead, different databases may be available in different domains based on the differences we introduce. This interdomain gateway discovery could be carried with the Telephony Routing Information Protocol (TRIP see [TRIP00]).

The endpoints can consult the gateway database with the Lightweight Directory Access Protocol (LDAP) protocol. This protocol allows accessing any directory organised in a special tree structure. Thus when LDAP is used, the gateway database must be organized belong the LDAP data model. To reiterate, location servers can use the TRIP protocol in order to build a database that the users can consult with LDAP. The purpose is to find the gateway that can terminate the call as efficiently as possible.

3.4.6 Address translation

In the PSTN telephone numbers are defined by the E.164 Recommendation from ITU-T (see [E.164]). A classical CPE, like a simple telephone, has been engineered in conformance with this recommendation. Unfortunately, it does not have the capability to define anything else other than digits (IP addresses for example). Thus a solution must be found to establish a phone-to-PC call. This solution consists of giving E.164 numbers to Internet telephony's users. Of course, these numbers must adhere to a special numbering plan that is distinct from the one used in the traditional network. As a consequence, a search-directory protocol, like LDAP can be used in order to translate the telephone number into an IP address.

From the IN's view, this solution has a great advantage: the directory may contain other information than just the IP address! Suppose that a user does not want to be disturbed during her or his meeting (between 2 p.m. and 4 p.m.). This information can be stored in the directory as well as her or his current IP address. If someone calls this user at 3 p.m., the call may be routed to her or his mailbox.

IP addresses are changing all the time: any user that accesses the Internet via a dial-up has almost never the same IP. Whatever the protocol used, it must allow frequent updates in the directory due to the volatility of current IP addresses. Moreover, it may provide enough performance in order that a query does not introduce a significant delay in the call establishment.

We end this section with another problem that occurred in this new environment. The majority of IP-networks that comprise the Internet are using the fourth version of IP. With the current status of the Internet, the scarcity of IPv4 addresses has become a really annoying problem[1]. The short-term solution to this problem is the IP Network Address Translator (NAT) defined in [RFC2663]. In a nutshell, such address translation allows hosts in a private network to transparently communicate with destinations on an external network and vice versa. With such a system, a company may have IP addresses that exist already in the Internet. As a result, call establishments must be dynamically modified in order to contact the correct IP. An Internet draft provides a solution for the SIP protocol (see [NAT00]).

3.5 Supporting IN services

In this section, we show how the existing IN services can be supported in this new architecture. These are, for instance, freephone numbers, existing call centers, etc. The SIP protocol, for instance, can provide a subset of these existing services, such as call-forwarding, follow-me, do-not-disturb, etc. Thus we provide here a high-level view of the provision of existing IN services. This view is sub-divided into two scenarios. First, all the services resides in the PSTN. Then, a more complicated one, services are implemented in the PSTN and on IP-hosts.

3.5.1 First scenario: reaching IN in the PSTN

As we have already stressed, users must have the ability to contact existing services, even if they are placing calls in the IP world. Figure 3.5 shows what happens when a user tries to contact a call center via a freephone number. This example utilizes the protocols we have seen in Figure 3.3. Thus, the user can utilize SIP or H.323 to ask for the establishment of the call. The softswitch can use the MEGACO protocol to control an MG which is available. Moreover it can use the SCTP protocol to transport SS7 signalling over IP to the SG.

[1] In fact, there are too many Internet users to give an IP address to each of them.

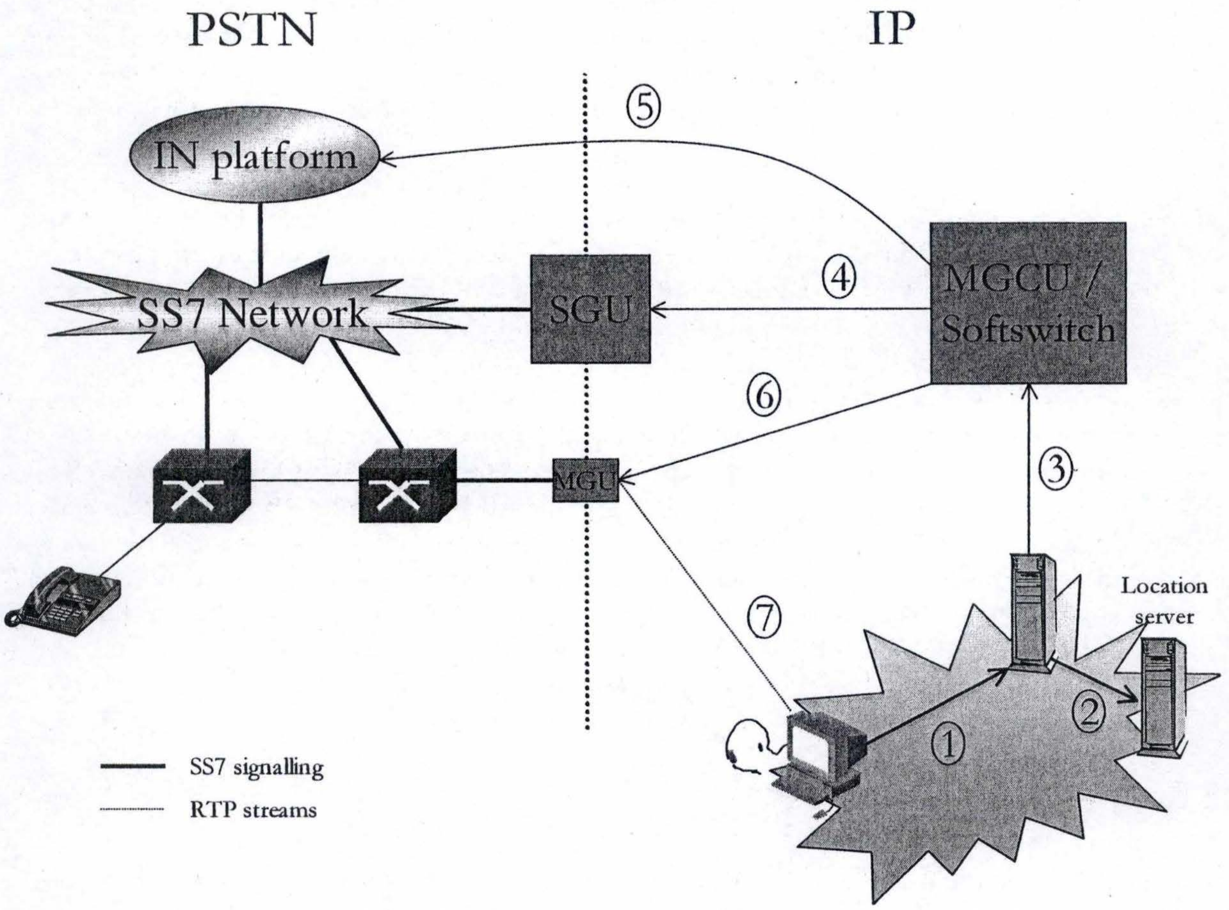


Figure 3.5 – First scenario example

In (1), the user asks for a call establishment. To do so, the protocol used (e.g. SIP or H.323) contacts the local server of this user. In (2), the local server asks the location server which gateway must manage this call. Since this user wants to contact a "user" in the PSTN, there is no problem of address translation. In (3) the request is forwarded to the specified gateway. At this point, two solutions exist.

The first is not described in the above figure. The softswitch, together with the SG function as shown in (4), simply contacts the nearest SSP. It implies that the whole IN process is performed in the PSTN. This could imply some problems would exist when controlling an MGU (for digit collections) from a traditional SSP, for instance.

The other is the one described in Figure 3.5. Since the softswitch is configured as an SSP, it may contact directly, as shown in (5), an SCP through an IP interface. To do so, the specified SCP must have an IP interface properly configured in addition to the SS7 interface to the traditional network. This is not restricting since an SCP is running on a general computer: an IP interface can be added easily. This solution has the drawback that the softswitch has to be updated like any other SSP when a new class of triggers is implemented, for instance. However, the interaction between MGUs and the SSP (here the softswitch) is much more efficient.

Remember the freephone example of chapter 2. The freephone number is translated back to the softswitch. Once the user is contacted, the softswitch is notified (since it initiates the call). In (6), it instructs the optimal MGU (e.g. by using MEGACO) to open an RTP stream with the user. In

(7), the call between the user and the specified freephone number (here a call center) is established.

Note that when the ISUP::SETUP is sent in (4), the voice circuit channel identifier is well-known and corresponds to a physical termination in a MGU. The choice of the optimal MGU is then taken in (4), which is then contacted by the MGC in (6).

The softswitch manages the call. If the call center asks for some digits, the originating switch (i.e. the softswitch) is instructed via the INAP protocol to do so. As a consequence, the softswitch uses MEGACO to order the MG to collect the digits. This example shows how the features are unbundled in the network. Digits collection over RTP streams has caused a lot of problems because RTP was not initially designed to carry critical voice patterns without specific RTP payloads. Hopefully, these payloads are well defined and supported now.

3.5.2 Second scenario: reaching IN in IP

In this scenario, the gateway supports the access to IN services that are residing in the IP network. Even if a web call center can have an IP address (or a SIP address for instance), it is important to support the billions of PSTN users. This can be done by providing a freephone number for this web call center.

The reader may suggest coming back to the first scenario, since this web call center has an E.164 address. Such a choice must be avoided. Indeed, this has two consequences. First, the IN platform is used by IP users and thus we do not have an NGN philosophy. In other words, we do not take advantage of the Internet architecture. More precisely, such an architecture allows storage of IP-specific information that could improve the service for IP-users. The second consequence is that the IN platform is only used to translate the freephone number into another E.164 address. After this information has come back from the PSTN, it is finally translated into an IP address. Such a mechanism could add a significant delay in the call establishment and it is probably more efficient to retrieve the information directly from the IP network. Figure 3.6 schematises an architecture that supports this scenario.

Steps (1) to (3) are exactly the same as the first scenario. This enables the end-user to access these services with excellent transparency. The major changes occur in the softswitch model. A number of modules have been added: these are the LDAP database, the local policy and the services modules. The first is used to look up information on the requested number. Since Internet telephony's CPEs have also an E.164 address but probably a different numbering plan, the softswitch can determine if the called party resides in the PSTN or in an IP network. The local policy is used to determine how this request has to be managed (e.g. billing, authorization, authentication, etc.). Finally, the services module allows the user to access the traditional IN platform as well as the set of services that we define in the next chapters (see these chapters for more details).

In (4), the softswitch consults its database and checks that the called party resides somewhere in the IP network. The database may contain other information that could be useful for the service logic of the softswitch. For instance, IN-like factors, such as the location of the calling party, can influence the decision. In (5) the softswitch uses the service module to determine where the call has to be routed and how. Then, in (6) it sends back the response to the local server for this user. Another solution is to connect the user directly to the called party, i.e. by proxying the request. In (7) the calling party and the called party are connected: this could be done by using H.323 or SIP.

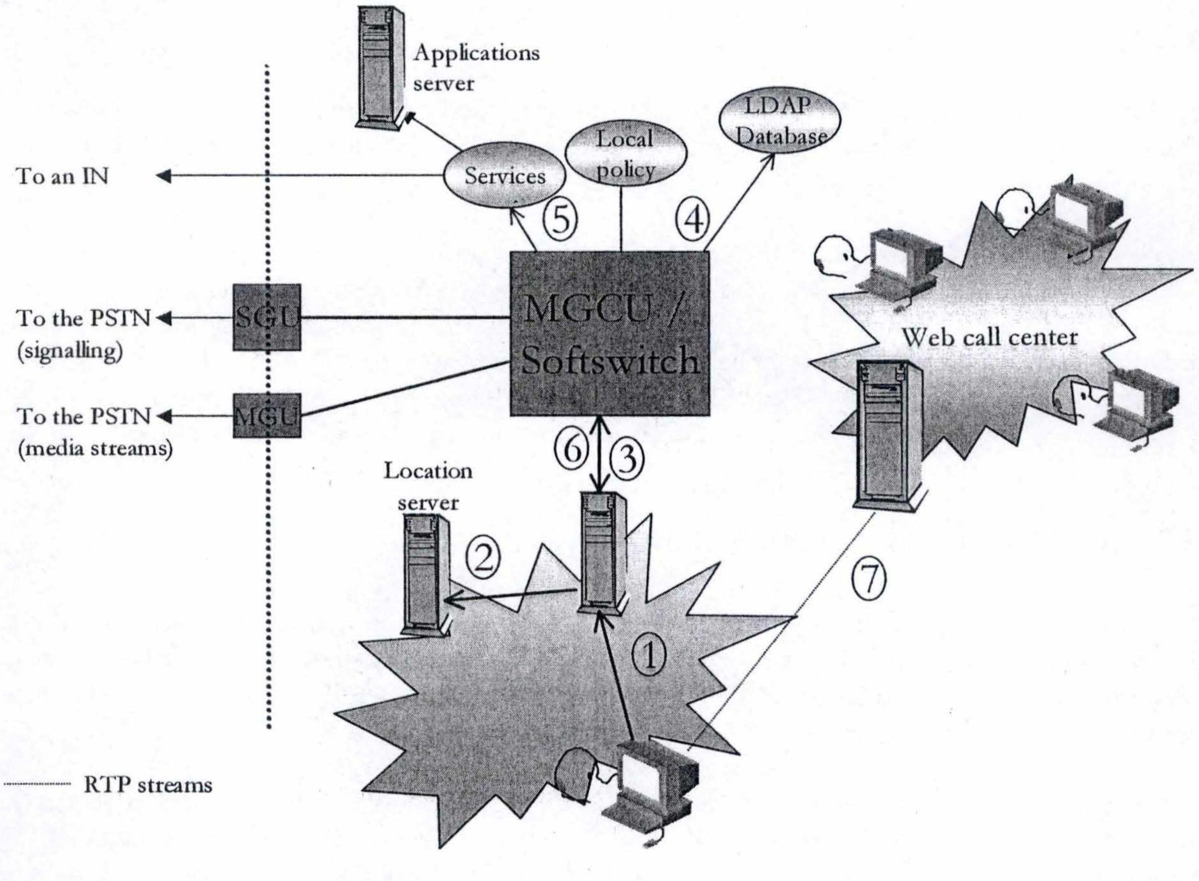


Figure 3.6 – Second scenario example

Pay attention to the fact that once the calling party has reached its destination, another service logic can take place. The local server of the web call center can decide which operator has to respond, based on the calling party's information.

3.6 Wrapping up and conclusion

In this chapter, we have seen how the Internet and the PSTN could be modified to interwork. On the one hand, the PSTN has to handle the dialup traffic problem. Moreover, the bandwidth available in the local loop could be improved, since new services and voice traffic are requiring higher bandwidths. On the other hand, the IP world has to provide a QoS level in order to support real-time traffic (e.g. telephony services). We have discussed possible implementations (MPLS, intserv, diffserv, ATM over IP) which are not exhaustive. Then we have lingered over the gateway concept. Keep in mind that the MGC-MG-SG model is still evolving and a lot of work has to be done to specify all the interfaces between the PSTN and IP networks. Finally, we have shown how existing services can be supported within the IP network and how new services can be hosted in IP.

In the next chapters, we show how the new generation of services can be programmed and supported. There we discuss in more details the services module of Figure 3.6.

Chapter 4

Providing Advanced Services

4.1 Introduction

The first three chapters we have already covered has shown us that the IN landscape is changing. In this chapter, we study how new and innovating services can be developed with this new environment. In this dissertation, we have already seen that carrying voice traffic on an IP network was not a straightforward task. In the same way, we have seen a number of problems concerning IN services, e.g. customisation and interworking between INs. Finally, we have decided, in appendix 5, to chose SIP as a signalling standard for Internet telephony.

Chapter 2 has introduced you to IN services: these are additional features associated with a voice call. Those that concern Internet services are so widely used today that we know we do not have to introduce them. The Internet has brought an incredible number of services and they are still growing today. With the network convergence we have studied so far, a number of opportunities can bring even more interesting services. For instance, the CPE is moving into a powerful multimedia computer. Such a move can simply delete the customisation problems that exist in traditional INs.

With the growth of the Internet, teleoperators must take advantage of the advisability of developing a new portfolio of services. This results first in what is called hybrid services. Like hybrid networks that we have extensively studied in chapter 3, hybrid services are services that use both the PSTN and the Internet. In this field, two working groups at the IETF come to the fore. First, the PSTN/Internet INTernetworking (PINT) working group (see [PTWG]) addresses connection arrangements through which Internet applications can request and enrich PSTN telephony services. In other words, services are initiated in the Internet and carried out in IN. The other one is the Service in the PSTN/IN Requesting InTernet Services (SPIRITS) working group (see [SPWG]) which addresses how services supported by IP network entities can be started from IN requests, i.e. in contrast to PINT. Coupled together, the PSTN and the Internet can complement each other quite effectively. For example taking the flexible customisation of the Internet and the powerful reliability of IN.

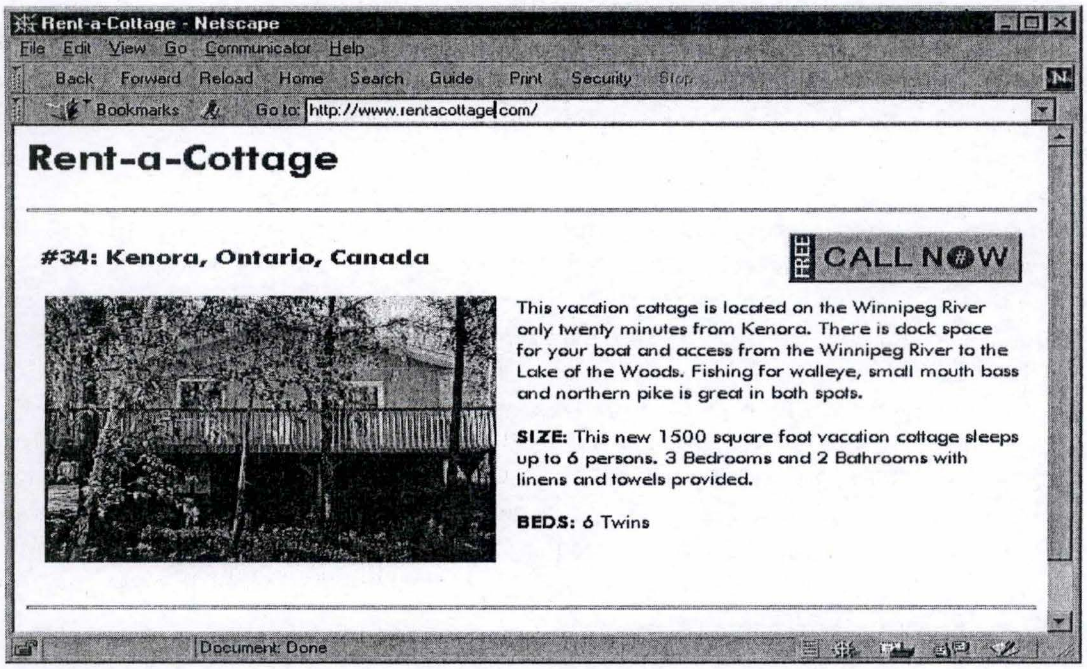
In this interworking model, there are more than phone-to-PC and PC-to-phone calls. There are also PC-to-PC calls and these should also provide additional features like IN services that the user can find in the PSTN. We have chosen SIP as the signalling standard for Internet telephony and thus we discuss only how SIP can be used to provide additional services. Indeed we discuss a way of programming advanced features with SIP, i.e. SIP CGI.

4.2 PINT

The PSTN/Internet internetworking (PINT) protocol is used for invoking certain telephone services from an IP network. These services are: making a phone call, sending and receiving faxes, and receiving content over the phone. The PINT protocol is an enhancement of the SIP and SDP protocols; it systematically uses the IN as the means of uniting the PSTN and the Internet. In this section we show how the PINT protocol is bringing network intelligence to the edge of the network. The survey we give here of PINT is based on [RFC2848].

4.2.1 Introduction

We show here a simple example of what PINT can offer as well as the basic concepts involved in the PINT protocol. The Request to Call service allows a user to send a request from an IP host which has the purpose of establishing a phone call between two parties. Figure 4.1 shows an example of this functionality (often called Click-to-Call). If the user wants to have more details on this cottage, (s)he has simply to click the "call now" button to establish a phone call with an operator of this company.


Rent-a-Cottage - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks
Go to: http://www.rentacottage.com/
Document: Done

Rent-a-Cottage

#34: Kenora, Ontario, Canada

FREE CALL NOW

This vacation cottage is located on the Winnipeg River only twenty minutes from Kenora. There is dock space for your boat and access from the Winnipeg River to the Lake of the Woods. Fishing for walleye, small mouth bass and northern pike is great in both spots.

SIZE: This new 1500 square foot vacation cottage sleeps up to 6 persons. 3 Bedrooms and 2 Bathrooms with linens and towels provided.

BEDS: 6 Twins

Figure 4.1 – A PINT-compliant website

In the next section devoted to SPIRITS, we show how this service can be enhanced and how PINT and SPIRITS can be combined to provide powerful services.

When the invocation of a PINT service is successful, the generic scenario is as follows: first, an IP host sends a request to a server on an IP network. Then the server relays the request into a telephone network. Finally the telephone network performs the requested call service. This scenario shows how PINT services are hybridised, i.e. involving two different networks. Indeed, a complete PINT transaction involves an IP network as well as the PSTN.

The PINT's scope in terms of services has been deliberately restricted to provide only a small number of services. The main reason for this is of course the security issues related to many services. These services, however, can be used as the building blocks for many other services. There is presently a proposal for conference calling, with which the PINT protocol could be modified to provide conference control (see [KFL00]).

The PINT protocol is an extension of SIP and SDP. A PINT client who wishes to invoke a service uses SIP in order to establish a session with the PINT server. The SIP invitation contains a SDP description of the media session involved with this service. Of course, the enhancements and additions specified by the PINT protocol are not intended to alter the core of SIP or SDP in any way.

The set of services proposed in the PINT protocol, called the PINT milestone services, are described in section 4.2.2. The PINT architecture is described in section 4.2.3. Then SIP extensions are described in section 4.2.4 while SDP ones are described in 4.2.5. We go on with some examples of PINT services and we conclude this part with the PINT status.

4.2.2 PINT milestone services

There are 3 main telephone services that a PINT client may invoke: Request to Call, Request to Fax Content and Request to Speak/Send/Play Content.

The Request to Call service has already been overviewed in 4.1.1. In this service, A's request is sent from an IP host that causes a phone call to be made. This call connects A with another party (for example an operator at a booking office).

In the Request to Fax Content service, A's request is sent from an IP host, the former causes a fax to be sent to a fax machine. Many ways are given to define the content to be faxed. It could be for instance a web page located somewhere in the network or the content of the SIP message itself. Moreover the content may be text or images. It should be noted that this service is not a Fax over IP service. Like all PINT services, the IP network is only used to request the service. The details of the fax transmission are managed by the telephone network.

In the Request to Speak/Send/Play Content, A's request is sent from an IP host, the former causes a phone call to be made to user A, and for some sort of content to be spoken out. This content may be appointed by an URL for example or be contained in the request itself. As for the Request to Fax service, the content may be text or some other data type. Again the details of the transmission are managed by the telephone network. This service is also called Request to Hear Content. The way the request is formulated is outside the scope of the PINT protocol as defined in [RFC2848].

4.2.3 PINT Architecture

According to [RFC2848], "PINT clients and servers are SIP clients and servers. SIP is used to carry the request over the IP network to the correct PINT server in a secure and reliable manner, and SDP is used to describe the telephone network session that is to be invoked or whose status is to be returned."

A PINT-compliant network uses a SIP proxy server and redirect servers to the features they offer as described in appendix 5. However, requests must reach a PINT server that is able to interact with the telephone network in order to provide the requested services. This server is called a PINT gateway. The PINT gateway relays received requests into the telephone system and receives acknowledgements of these relayed requests. It appears to a SIP system as a SIP User Agent Server and represents in the PINT infrastructure an entire telephone network infrastructure that can provide a set of telephone network services.

A PINT system is composed of 3 basics elements[1]: PINT client (SIP User Agent Client), PINT gateway (SIP User Agent Server) and executive system. These notions are represented in Figure 4.2 (see Figure 1 in [RFC2848]).

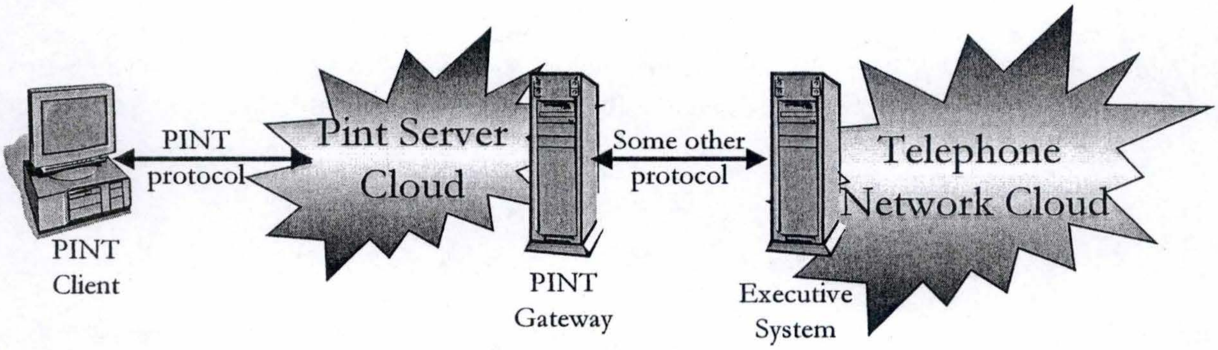


Figure 4.2 – PINT functional architecture

"The system of PINT servers is represented as a cloud to emphasise that a single PINT request might pass through a series of location servers, proxy servers, and redirect servers, before finally reaching the correct PINT gateway that can actually process the request by passing it to the Telephone Network Cloud."

The ITU-T is responsible for the IN part of PINT. Topics concerning PINT are under the auspice of Study Group 11, which is specifying requirements for a functional architecture that supports IN and the Internet interworking. The results of this work are due to be included in the CS-4 documentation.

PINT requests do not specify too precisely the exact telephone-side service. Operational details of individual events within the telephone network that execute the request are outside the scope of PINT. Assume that a user clicks on a web page to contact someone in a call center (for instance +32-800-12345). In the telephone network, IN is used to decide exactly which of the hundred agents in the call center will answer the call. Once this choice is made, there may be once again different scenarios. Thus the scope of PINT is intentionally restricted to the PINT server cloud for that very reason. However this fact does not prevent one expressing additional preference for a PINT service. The SDP payload may, for example, contain the language preference of the user for the Request to Hear Content service.

[1] To this we must also add SIP proxy servers and redirect servers.

4.2.4 PINT extensions to SIP

SIP is used to carry the request for telephone service from the PINT client to the PINT gateway. This request may include a telephone number if needed. The extensions and enhancements of SIP are as follows:

1) The multipart MIME payloads
2) Mandatory support for "Warning:" headers
3) The SUBSCRIBE and NOTIFY, and UNSUBSCRIBE requests
4) Require: headers
5) A format for PINT URLS within a PINT request
6) Telephone Network Parameters within PINT URLs

We have seen that PINT request for a Request to Fax for instance can convey the content that has to be faxed. The support of MIME payloads in SIP allows sending data in more than one part. This could be useful for sending messages where the length is limited.

A PINT Server must support the warning header[1] in order to notify the lack of support for individual PINT features. For example it may notify the PINT client that it does not support the type of data specified in a Request to Call service.

When a PINT server agrees to send a fax on a particular fax machine, the transaction may fail after part of the fax is sent. Therefore, PINT clients must have the ability to receive information about the status of the call. For instance, PINT clients may receive the number of pages that have already been correctly transmitted. To do so, three new methods have been added to SIP. When a SUBSCRIBE request is sent to a PINT Server, it indicates that a user wishes to receive information about the status of a service session. During the subscription period, the PINT gateway may send an asynchronous NOTIFY request to the entity that has subscribed to the monitoring of the session. NOTIFY messages may be sent as a result of any change in the status of the service session. UNSUBSCRIBE is used to end the monitoring of a service session.

PINT clients use the Require header to signal to the PINT server that a certain PINT extension of SIP is required. PINT version 1.0 defines two strings: "org.ietf.sip.subscribe" means that the PINT server can fulfil SUBSCRIBE request and associated methods; "org.ietf.sdp.require" means that the PINT server understands the require header.

In appendix 5, we have seen that Bob requests a call to be established between himself and Alice by using the SIP protocol and the SIP address of Alice (sip:alice@example.com). The PINT URL within a PINT request has 3 improvements. First, when used in the "Request-URI"[2], it could be useful to indicate which service is requested within this URL in a standardized way. This could be useful because it may not be possible to use SDP information to route the request if it is encrypted for instance. Thus the following standardisation applies: R2C is used for Request to Call, R2F for Request to Fax and R2HC for Request to Hear Content (Request to Speak/Send/Play Content). Second, the host portion of a sip URL contains the domain of the PINT service provider. Third, a new URL parameter is defined to be tsp (Telephone Service Provider). This can be used to indicate the actual TSP to be used to fulfil the PINT request.

[1] The SIP warning header is used to carry additional information about the status of the response (see [RFC2543, p 60]).

[2] The "Request-URI" field indicates the user or service which this request is being addressed to. Unlike the "To" field, the "Request-URI" may be re-written by proxies.

```
INVITE sip:R2C@pint.mytelco.com;tsp=pbx29.mytelco.com SIP/2.0
```

We close this section by discussing telephone network parameters within PINT URLs. Because PINT is based on SIP (and SDP) any legal SIP URL may appear as a PINT URL within the "To" or "Request-URI" headers of a PINT request. If the address is a telephone address, it may be necessary (see section 3.4.3 of [RFC2848]) to include more information in order to identify correctly the remote telephone terminal or service.

4.2.5 PINT extensions to SDP

The SDP payload contains a description of the particular telephone network session that the requestor wishes to occur in the PSTN. The extensions and enhancements of SDP are as follows:

1) A new network type "TN" and address types "RFC2543" and "X-..."
2) New media types "text", "image", and "application", new protocol transport keywords "voice", "fax" and "pager" and the associated format types and attribute tags
3) New format specific attributes for included content data
4) New attribute tags, used to pass information to the telephone network
5) A new attribute tag "require", used by a client to indicate that some attribute is required to be supported in the server

We do not provide a survey of these here, the interested reader may consult [RFC2848] for more details. However, it is important to say a few words about the attributes tags that are used in PINT to pass information to the telephone network. These tags can be understood only by some entity in the "Telephone Network Cloud".

Phone-context attribute is one of these tags. This attribute is specified to enable "remote local dialling". For instance, you may want to contact your telecommunication provider to report a telephone failure. Assume that this number is 555. If you use the Request to Call service and this uses another operator's network, it will not connect to the helpdesk and +32 555 will not normally succeed. Such numbers are called Network-Specific Service Numbers and the phone-context attribute is used to provide the local context needed to connect to such numbers. Another tags are the ITU-T CalledPartyAddress attributes parameters, which are used to convey further parameters relating to the terminal address to the Telephone Network Cloud.

4.2.6 Request to Call example

We go on with the example described in Figure 4.1. Figure 4.3 shows what happens when user A clicks on the "call now" button.

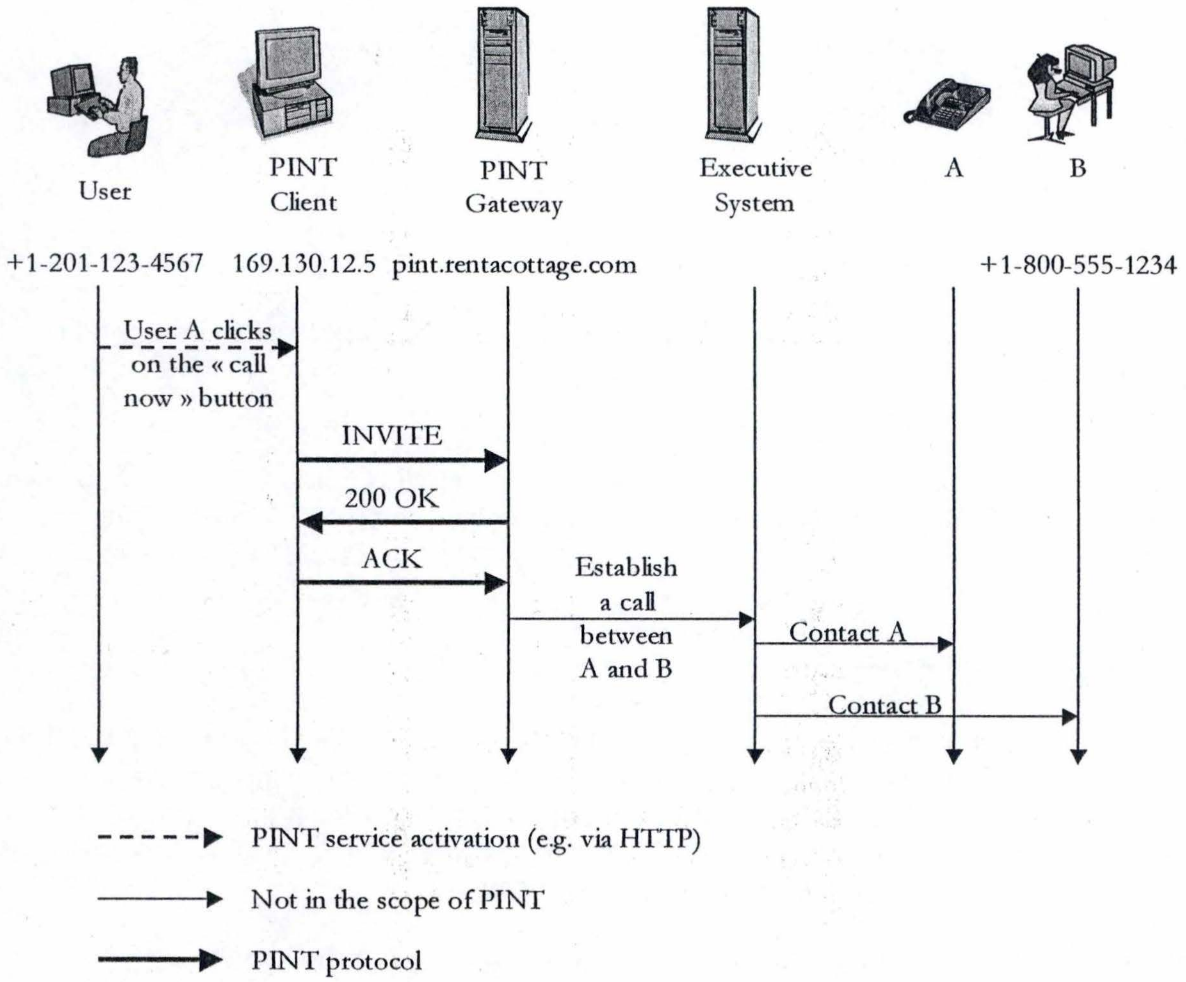


Figure 4.3 – Click-to-Call example

In the figure above, B is supposed to be an operator at the booking company. If this company has a call center managed by an IN, the IN will take over all the actions needed to establish the call between A and B. Please remember that these actions as well as the way the request is formulated are outside the scope of PINT. In our example, the request is formulated via a button on a web site but it is not the sole solution. Below is the INVITE message that is sent between the PINT client and the PINT gateway in Figure 4.3.

```
INVITE sip:R2C@pint.rentacottage.com  SIP/2.0
Via: SIP/2.0/UDP 169.130.12.5
From: sip:anon-1827631872@chinet.net
To: sip:+1-800-555-1234@iron.org;user=phone
Call-ID: 19971205T234505.56.78@pager.com
CSeq: 1 INVITE
Subject: Additional information about cottage #34
Content-type: application/sdp
Content-Length: 174
```

```
v=0
o=- 2353687637 2353687637 IN IP4 169.130.12.15
s=R2C
i=Rent-a-Cottage-#34
e=anon-1827631872@chinet.net
t=2353687637 0
m=audio 1  voice -
c=TN  RFC2543  +1-201-123-4567
```

The "c" field of the SDP content contains the telephone number of A. The subject "Additional information about cottage #34" may be added automatically by the website based on the page the user was surfing to.

This service is also called Click-to-Call-Back since the phone call is not immediately established. Indeed, the user simply requests a phone call from the booking company but he does not have to call, the company contacts him directly.

4.2.7 Request to Fax example

In fact, the Request to Fax service contains two basic services: the Click-to-Fax service and the Click-to-Fax-Back service. In the first, it enables the user to request that a fax be sent to a particular fax machine. In the latter, a user can request that a fax be sent to her or him. For instance, (s)he may want to retrieve additional information about the cottage and asks the company to send him or her a fax containing this information.

In figure 4.4 we provide an example of Click-to-Fax service. We also show in this figure how the SUBSCRIBE method can be used. Assume the user wants to fax, to a particular fax machine, a web page. This web page is contained somewhere in the network (for instance: http://www.rentacottage.com/info/#34.html). The content of this web page is 5 A4 pages long. The user may want to know the status of the service session, for example how many pages have already been faxed successfully. The PINT client may fulfil this role by using the SUBSCRIBE method. Once the PINT gateway sends it a notification (by using the NOTIFY method), the PINT client may refresh a page giving the status of the fax transmission. An example of such page is schematised in Figure 4.4.

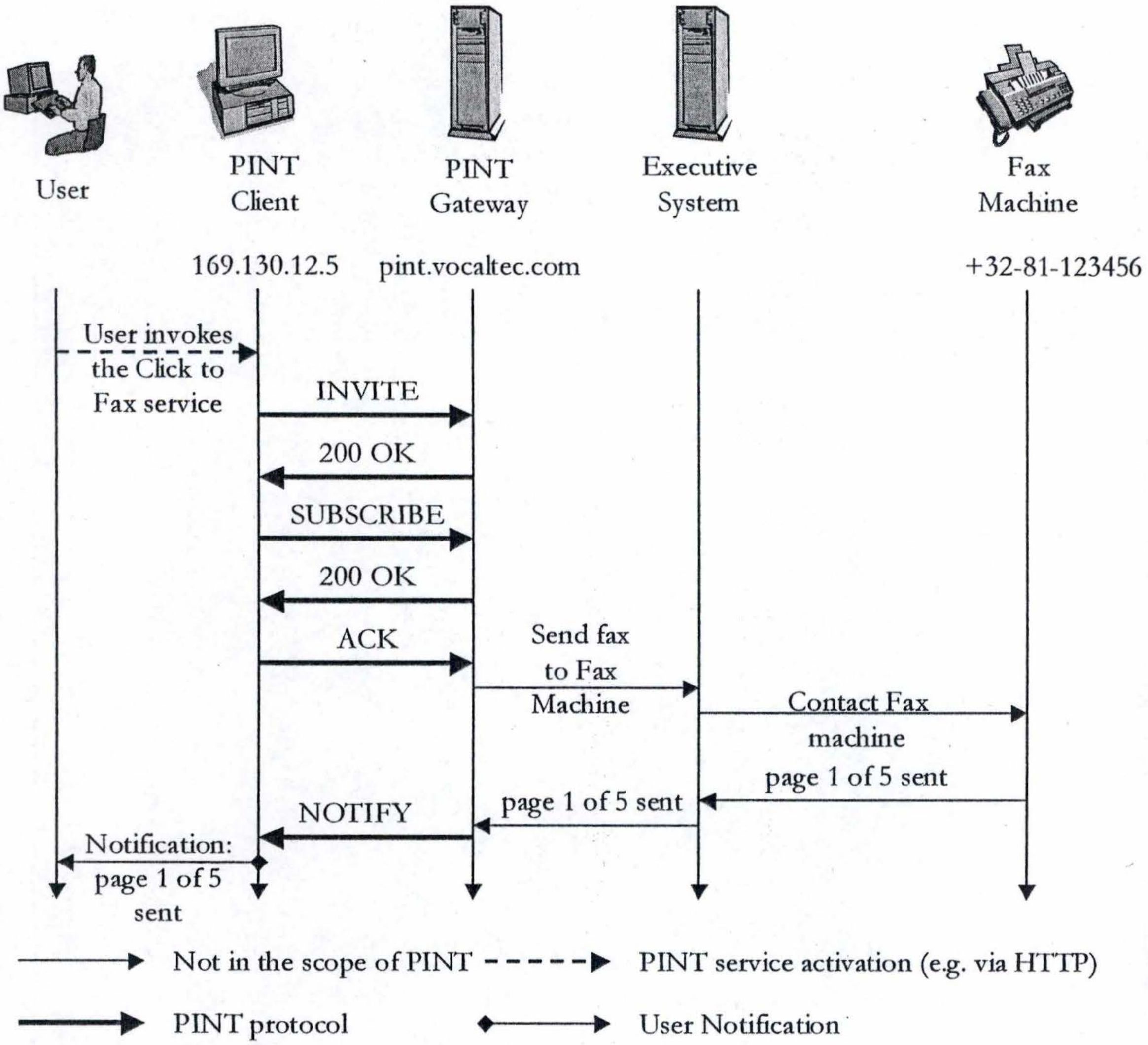


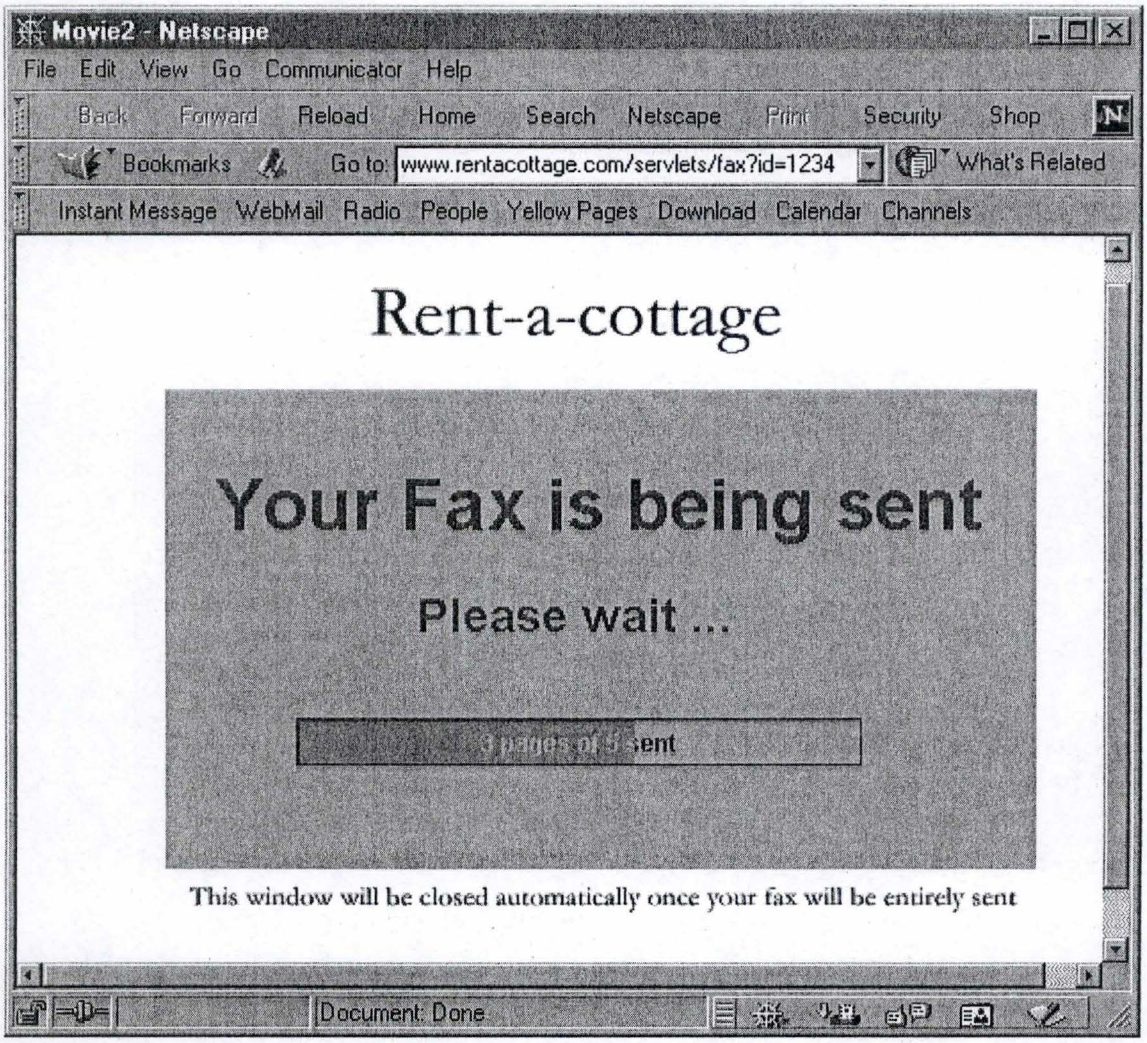
Figure 4.4 – Click-to-Fax example with the SUBSCRIBE method

Below is the INVITE message that is sent between the PINT client and the PINT gateway in Figure 4.4.

```
INVITE sip:faxserver@pint.vocaltec.com  SIP/2.0
Via: SIP/2.0/UDP 169.130.12.5
From: sip:anon-1827316278@chinet.net
To: sip:faxserver@pint.vocaltec.com
Call-ID: 19971205T234505.66.79@chinet.net
CSeq: 1 INVITE
Content-type: application/sdp
Content-Length: 193

v=0
o=- 2353687700 2353687700 IN IP4 169.130.12.15
s=faxserver
e=anon-1827316278@chinet.net
m=application 1 fax URI
c= TN  RFC2543  +32-81-123456
a=fmtp:URI uri: http://www.rentacottage.com/info/#34.html
```

Figure 4.5 shows how dynamic content (via a Java applet for example) can notify the user of the status of the service session.



Movie2 - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop
Bookmarks Go to: www.rentacottage.com/servlets/fax?id=1234 What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Document: Done

Rent-a-cottage

Your Fax is being sent

Please wait ...

3 pages of 5 sent

This window will be closed automatically once your fax will be entirely sent

Figure 4.5 – User notification example

4.2.8 PINT status and conclusion

RFC 2848 describing the PINT protocol is a standard track and lots of Internet drafts have been written in relation with this protocol. Implementations of PINT already exist, eDial for example[1]. But until the work from the ITU-T describing the IN part of PINT is released, it is difficult to foresee the PINT's future.

The PINT protocol represents hybrid services and thus constitutes a mid-term solution. The way towards Next Generation Networks is long and such solutions can provide a lot of experience for the future. Extensions to conference calling are described in [SLU00] and service examples in [SIPEX01].

[1] See http://www.edial.com.

4.3 SPIRITS

The service in the PSTN/IN requesting Internet services (SPIRITS) working group addresses how services supported by IP network entities can be started from IN requests. This section exposes the strong relationships between PINT and SPIRITS.

4.3.1 Introduction

In this introduction we describe an example of SPIRITS services, which are those originating in the PSTN and necessitating the interactions between the PSTN and the Internet. The most important one is the Internet Call Waiting (ICW) service. Afterwards, we discuss how such services can be implemented.

A lot of people are using their telephone line to access the Internet (through a dial-up connection to their ISP). While they are connected to the Internet, there are often blocking their line for incoming and outgoing calls. At present, the solutions are to invest in a second line (e.g. ISDN) or subscribe to an ADSL line (where available), which may be too expensive.

The ICW service allows subscribers to be reachable for incoming telephone calls while surfing on the Internet. For instance, an incoming call can be routed to a mailbox, a second line, a VoIP connection or the actual line. It means that the service performs automatic disconnection from the ISP during the call and re-establishes it when the call is completed.

PINT is strongly associated with SPIRITS service. In the ICW case, PINT is used to perform registration, i.e. customisation of parameters related to the service. Once the user has subscribed to the ICW service while surfing, any call attempt is delivered depending on the customisation (s)he has performed previously.

Other SPIRITS services exist: Caller-ID Delivery and Internet Call Forwarding. Caller-ID delivery allows the subscriber to see the caller's number or name or both while being connected to the Internet. The Internet call forwarding service allows a service subscriber to forward an incoming call to another telephone number while being connected to the Internet. It should be noted that if the subscriber has only one telephone line and is using this line for the Internet connection, then these two services are a subset of the ICW service. It is time now to discover how such a service can be implemented.

4.3.2 SPIRITS architecture

The architecture we present here is based on [SFLW01]. In the same time, it shows how PINT and SPIRITS are related to each other. Figure 4.6 summarises the entities involved in the SPIRITS architecture.

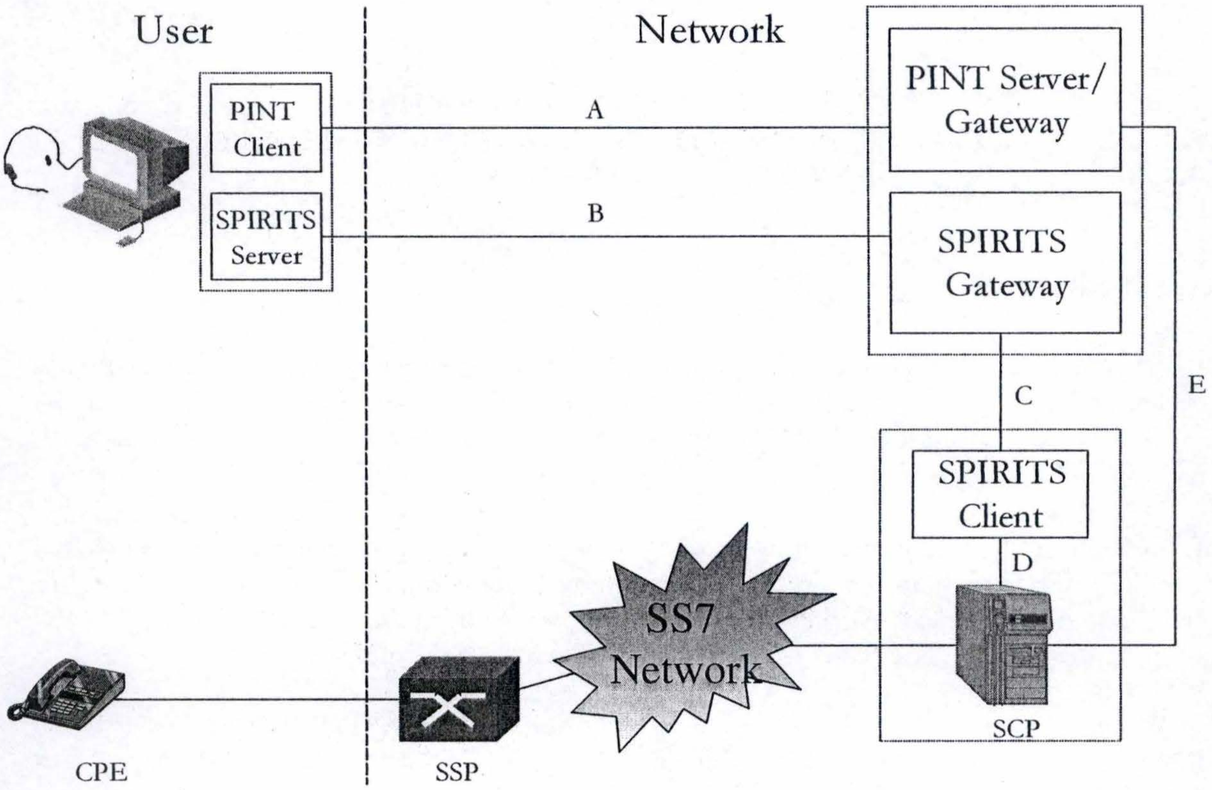


Figure 4.6 – SPIRITS architecture

The entities and functions involved in the SPIRITS architecture are:

- Service Control Function (SCF), which executes service logic, interacts with the entities in the IP domain (e.g., the SPIRITS Gateway and PINT Server) through the SPIRITS Client, and instructs the switches on how to complete a call
- Service Switching Function (SSF), which is responsible for the recognition of IN triggers and interactions with the SCF
- SPIRITS Client, which is responsible for receiving PSTN requests from the SCF as well as sending back responses
- PINT Server, which receives PINT requests from the PINT Client and relays them to the PSTN for execution over the E interface
- SPIRITS Gateway, which is co-located with the PINT Server or PINT Gateway (or both when they are co-located as assumed here for simplicity) and serves as an intermediary between the SPIRITS Server and SPRITS Client via the B and C interfaces, respectively
- PINT Client, which resides in the subscriber's IP host and is responsible for initiating PINT requests, which are sent to the PINT server over the A interface
- SPIRITS Server, which terminates PSTN requests and is responsible for all interactions e.g., incoming call notification and relaying the call treatment) between the subscriber and the SPIRITS Gateway

4.3.3 Internet Call Waiting: a SPIRITS service example

In this section, we illustrate the concepts seen previously with the ICW service. [RFC2995] presents 4 pre-SPIRITS implementations of PSTN-initiated services and in particular the ICW service. The interested reader may consult it to have a concrete view of how such an architecture can be implemented. The reader should note that the example we propose here is informal; it represents one of the possible implementations of the ICW service.

The PINT server/client are used here for registration purpose. When connection to the Internet is made, the user can register to the ICW service. The PINT protocol is used to invoke an IN service which configure the current status of this user.

Suppose that A initiates a call to B. B is currently surfing on the Internet and has registered to the ICW service. The task of the SSP is the same as the one we have studied in chapter 2. When detecting that the called party is busy, it sends a query to the SCP and waits for further instructions from the SCP. When triggered, the SCP launches the ICW service. The service logic governs the notification of a waiting call to an online ICW subscriber and the features of the call. To do so, it may instruct the SPIRITS client to perform some other actions.

Finally the SPIRITS client/server are used to decide the future of the call depending on the registration that has previously been done. These two entities may use the SIP protocol to communicate. In addition, the SPIRITS server may monitor the connection status of the registered ICW Clients. Thus, it may deactivate the ICW service for a client as soon as it detects that (s)he has deactivated the ICW service or terminated the Internet connection.

4.3.4 SPIRITS status and conclusion

It should be clear to you now that PINT and SPIRITS form a whole solution. For instance, the ICW service can be used after a Click-to-Call service. The user asks for more information on a cottage, continues surfing and is notified of an incoming call afterwards. Even if PINT and SPIRITS are complementary, work on SPIRITS is only at the premise of a final solution. The 4 implementations described in [RFC2995] are incompatible with each other. We provide here the conclusion of this document to illustrate this [RFC2995, p 40]: "It is clear that not all pre-SPIRITS implementations inter-operate with each other. It is also clear that not all SIP-based implementations inter-operate with each other given that they do not support the same version of SIP. It is a task of the SPIRITS Working Group to define the inter-networking interfaces that will support inter-operation of the future implementations of SPIRITS services."

These two protocols give a very good idea of the new generation of services that the network convergence between the PSTN and IP networks can offer. This solution is useful in a hybrid environment and shall most probably provide a lot of experience for the Next Generation Networks.

4.4 Programming advanced features with SIP

We have already seen that Internet telephony has the ability to contact IN networks to provide 800 number services, prepaid, etc. With a powerful signaling protocol like SIP, such services can be enhanced by interaction with e-mail, web, etc. Moreover, a new generation of services can be developed. In this section, we discuss how SIP can be used to fulfill these roles. First we give the basic requirements needed to perform this task. Then we discuss a solution that is already used worldwide in the Internet: Common Gateway Interface (CGI).

4.4.1 A generic model for programming SIP services

[RLS99] is an excellent document that provides a generic view of the requirements needed to program services with SIP. Figure 4.7 shows the model proposed in this document.

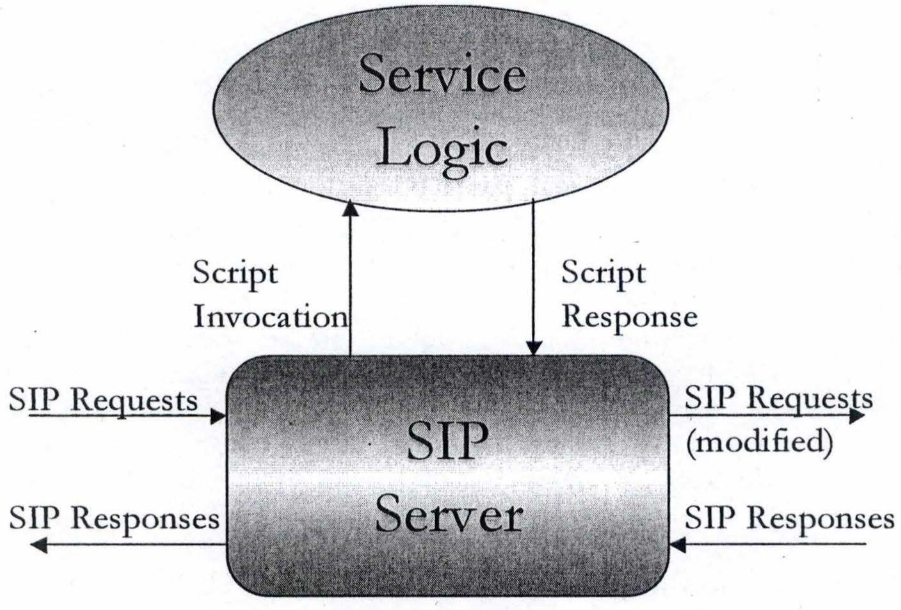


Figure 4.7 – Model for programming SIP services

The concept of separating service logic from the server is clearly an IN concept. This service logic is mainly in relation with SIP proxies. For example, the service logic can modify a request based on local policies or various factors. Moreover, it can be triggered by events and instruct the SIP server to generate new requests and responses. The service logic may also be implemented in SIP UAS (User Agent Server) but, "since user agents are usually owned by end users, not network service providers, providing logic for them is a different problem" [RLS99, p 5].

The applications of this concept are very wide. Web call centers can be enhanced with this system. The call center can, like classical call centers, have a unique number. The aim of the service logic is to balance calls between the operators. This balancing can be based on the caller, the time of the day, the location, the subject of the call. The SIP CGI that we discuss in the next section is clearly a good solution to provide this.

4.4.2 SIP CGI

The Common Gateway Interface (CGI) is an interface to the Web server that enables programmers to extend the server's functionality. Using CGI, (s)he can interact with users who access the site. On a theoretical level, CGI enables one to extend the capability of the server to parse (interpret) input from the browser and return information based on user input. On a practical level, CGI is an interface that enables the programmer to write programs that can easily communicate with the server. If you are unfamiliar to CGI, we refer you to [CGI]. Figure 4.8 shows a "Hello World!" CGI in perl:

```
#!/usr/local/bin/perl

print "Content-Type: text/html\n\n";

print "<html> <head>\n";
print "<title>Hello, world!</title>";
print "</head>\n";
print "<body>\n";
print "<h1>Hello, world!</h1>\n";
print "</body> </html>\n";
```

Figure 4.8 – Hello World! example

SIP relies heavily on HTTP, inheriting its client-server interaction and much of its syntax and semantics. For this reason, the web service creation environments, and CGI in particular, seem attractive as starting points for developing SIP based service creation environments. SIP CGI is defined in an informational RFC (see [RFC3050]). CGI has a number of strengths, which make it attractive as an environment for creating SIP services. Some of these strengths are:

- Language independence: CGI works with a lot of languages (perl, C, etc) as long as they support access to environments variables
- Exposes all headers: CGI exposes the content of all the headers in an HTTP request to the CGI application. For SIP CGI, it is very important since the script may alter the request to perform the selected service.
- Creation of responses: CGI is advantageous in that it can create all parts of a response. Since SIP CGI is not focused on the body, it is important to be able to generate all aspects of a response.

According to [RFC3050], an HTTP server is mainly concerned about the generation of responses. A SIP server is generally concerned about performing four basic operations:

- Proxying of Requests: Receiving a request, adding or modifying any of the headers, deciding on a set of servers to forward the request to, and forwarding it to them
- Returning Responses: Receiving a response, adding or modifying any of the headers, and passing the response towards the client
- Generating Requests: Creating a new request, originating at the server, placing headers and a body into the message, and sending it to a server
- Generation of Responses: Receiving a request, generating a response to it, and sending it back to the client

Moreover, there is a strong difference between HTTP CGI and SIP CGI: persistence. Whereas a classical CGI is executed once for each request, a SIP CGI script must maintain control somehow over numerous events. As you may recall, a number of messages can be exchanged between SIP entities in order to establish a SIP session. Thus, the SIP CGI model must be persistent. The solution proposed in [RFC3050, p. 7] is as follows: "State is maintained by allowing the CGI to return an opaque token to the server. When the CGI script is called again for the same transaction, this token is passed back to the CGI script. When called for a new transaction, no token is passed.".

The way SIP CGI can be implemented to provide SIP services is not described further here. The interested reader is invited to consult [RFC3050] for more details. Anyway, one should not forget that, at the time of writing, this proposition is in informational stage. Thus the development of SIP CGI should be taken with care since it is not widely deployed.

4.4.3 SIP CGI example

You may guess the advantages of a Web call center compared to a traditional one. A user is surfing on a commercial web site and wants to have more information about a product. The ability to have an operator that can respond to the user is very interesting. Moreover, since the parties are web-enabled, the operator can send to the user web pages containing further information while talking to him or her! In the PSTN, such a call center has most of the times a unique free phone number and the Intelligent Network is used to balance the calls among the operators based on some algorithms. In the architecture we described so far, web call centers can be implemented with SIP and SIP CGI scripts. Figure 4.9 below describes this. Assume that the user wants more information on cottage #34, as in the PINT example (see Figure 4.1). The web site provides a sip address: sip:info@rentacottage.com. This address has exactly the same purpose than a freephone number in the PSTN.

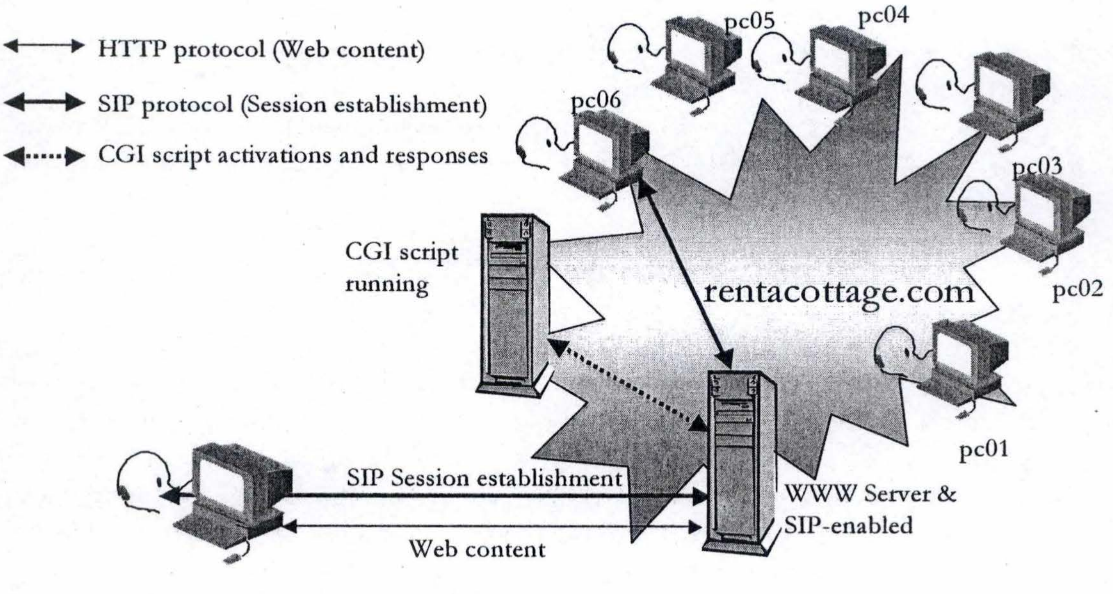

HTTP protocol (Web content)
SIP protocol (Session establishment)
CGI script activations and responses
pc05
pc04
pc06
pc03
pc02
pc01
CGI script running
rentacottage.com
SIP Session establishment
Web content
WWW Server & SIP-enabled

Figure 4.9 – Web call center using SIP CGI

Here is an example of INVITE message:

```
INVITE sip:info@rentacottage.com SIP/2.0
Via: SIP/2.0/UDP 169.12.34.45
Subject: (no subject)
From: sip:john.smith@example.com
To: info@rentacottage.com
Call-ID: 24684864864.44.44.79@example.com
CSeq: 1 INVITE
Content-type: application/sdp
Content-Length: 193
```

Once this request is received at the rentacottage domain, it can trigger the execution of a CGI script. The purpose of this script is to proxy the request to an operator which is available at the moment, for instance. Furthermore, any other logic can be implemented to proxy the request. For instance, the script knows that the user was surfing on the cottage #34. It can thus, based on this information, proxy this request to a specialist of cottages located in Canada. The script can also change the subject line since the user has not always enough imagination to write proper subject lines. Here is what the output of the script can look like:

```
CGI-PROXY-REQUEST sip:oper1_ca@pc06.rentacottage.com SIP/2.0
Subject: COTTAGE #34 Information request

CGI-SCRIPT-COOKIE abc-tervc SIP/2.0
```

The last line is used to enable the persistent model we have discussed above. If other messages are exchanged to establish this session and if a CGI script is required, the server will pass the specified token to the script, allowing it to re-establish the environment. The two first lines changed the original request this way:

```
INVITE sip:oper1_ca@pc06.rentacottage.com SIP/2.0
Via: SIP/2.0/UDP 169.12.34.45
Subject: COTTAGE #34 Information request
From: sip:john.smith@example.com
To: info@rentacottage.com
Call-ID: 24684864864.44.44.79@example.com
CSeq: 1 INVITE
Content-type: application/sdp
Content-Length: 193
```

You can see now that a specialist is called and the subject line has been changed to allow the operator to know which cottage the customer needs for additional information. Many more examples can be implemented with SIP CGI and the interested reader may consult the informational RFC for more details.

4.4.4 Evaluation and conclusion

The applications that can be developed with SIP deserve all our attention. We have seen SIP CGI and the web call center has shown us that CGI can be easily extended to provide telephone services. Another technology has always been related to CGI: servlet. Servlets are the counterpart of applets and are generally executed in response to an invocation from an HTML page. Servlets perform the same work as CGI but are more efficient because they don't create new processes every time they execute, unlike CGI. Since CGI and servlet have always been strongly linked, it is logical that an API has been written to support SIP services invocation within a servlet. This is done with the sipplet, a SIP-compliant servlet. The interested reader may find additional information in [DPJ00].

As you may have noticed, we have left the hybrid world to enter what is probably going to be the Next Generation Networks. Indeed, the integration with the Internet is as strong as PINT/SPIRITS protocols but the notion of hybrid networks has disappeared. Many more examples of application can be developed with the SIP functionalities. [DOY00] is a presentation showing interesting examples such as: integration of SIP and WAP (Wireless Application Protocol) and third-party applications (delivery tracking, call profiles).

Chapter 5

Open API Networks

5.1 Introduction

In this chapter, we discuss the ultimate way towards NGNs: open API networks. Service providers are searching for a way to rapidly provide services. An in vogue demand today is to have network products with open Application Programming Interfaces (APIs). These APIs allow the network to be augmented or modified to rapidly introduce new services consistent with the Internet model.

According to [LUC00a], technology and market advances have motivated this approach. For instance:

- Distributed systems provide operational efficiencies, improved reliability and availability over vertically integrated systems (such as the one existing in the PSTN)
- Object-oriented programming such as Java for instance provides software reuse and abstraction

A new business model appears with such a programmable network. "This model involves opening up networks to third-application providers. In this model the network provides services like call control, addressing, location, billing and notification. Third-party application, [...], access these network services." [LUC00, p 186]. The challenge for network operators is to allow third-party applications while protecting the network from harm.

In this field, two industry initiatives come to the fore: these are the Parlay consortium and the Java APIs for Integrated Networks (JAIN) initiatives. During the time of this writing, Parlay and JAIN have joined forces. We also describe the results of this alliance. Then we show how the web call center that we have discussed in the previous chapters can be implemented using the JAIN Parlay technology. We end this chapter by comparing traditional IN with the JAIN Parlay technology.

5.2 Parlay

The Parlay specification (see [PARLAY]) gives access to network information and allows control to a selected range of network capabilities. First of all, we give some definitions and objectives. Then we linger over the Parlay framework and Parlay services, illustrated by some sequence diagrams.

5.2.1 Parlay: definition, actors and objectives

The Parlay specification is a high-level specification written in UML, the Unified Modelling Language (UML). [BGD00, p 137] gives an excellent definition of this API: "The Parlay API defines a set of technology-independent interfaces that specify methods, events, parameters, and their semantics to allow external (untrusted third parties) and internal (traditional network operators) application creators the control over core network resources and capabilities."

The specification is completely technology-independent to allow as many market players as possible to offer advanced telecommunications services. The Parlay API consists of two categories of interfaces [PAR00, p 2]:

- Service Interfaces: these offer applications access to a range of network capabilities and information,
- Framework Interfaces: these provide the supporting capabilities necessary for the Service Interfaces to be secure, open and manageable.

Functions provided by the service interfaces allow access to traditional network capabilities such as call control, call management, messaging and user interaction. Functions provided via the framework interfaces are service registration, service subscription, service discovery, authentication, authorization and integrity management.

The Parlay API is composed of four main actors, i.e. the enterprise operator, the client application, the service provider and the framework operator. The first two are grouped as clients while the last two are grouped as providers. The enterprise operator has a network domain composed of her or his client applications. The same goes for the service supplier and her or his services. We find here a strong relationship with the IN actors covered in chapter 2. This comparison is summarized in Table 5.1.

Parlay actors	IN actors
Enterprise operator	Service subscriber
Client application	Service user
Service provider	Service provider
Framework operator	Network operator

Table 5.1 – Comparison between Parlay actors and IN actors

The API defines object-oriented interfaces on both the network and client application sides in the form of network interfaces and client application callback interfaces. Callback interfaces allow the server to interact with the client. In the authentication process for instance, the client gives to the framework a callback interface. This interface is used afterwards to retrieve authentication-related information about this client. Current version of the Parlay APIs is 2.1 but they are evolving and the version 3.0 is expected to be released in mid-2001.

5.2.2 Parlay Framework architecture

The purpose of the Parlay framework is to provide the needed features for the interface between clients and services to be secure, open and manageable. Figure 5.1 summarizes the actors of a Parlay-system and the APIs that compose the framework.

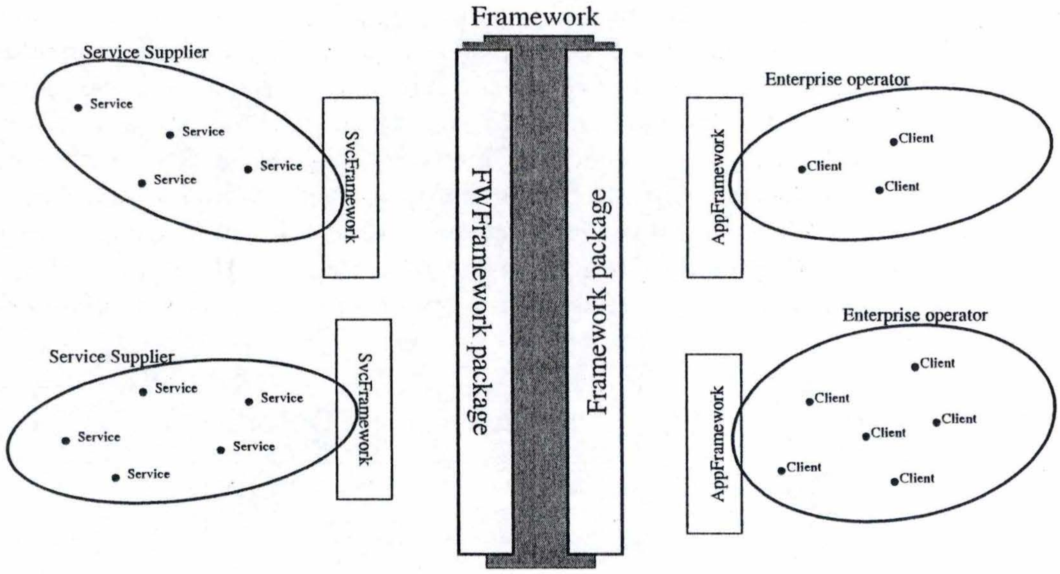


Figure 5.1 – Parlay framework APIs

The framework describes the information flows necessary between the service supplier (i.e. a particular service) and the domain of the enterprise operator (e.g. a particular client application). From the enterprise operator's view, the Parlay framework should be considered as a secure access to a wide range of services. From the service supplier's view, the framework should be considered as a context to outsource network services safely. As shown in Figure 5.1, APIs are regrouped by actors. The FWFramewok and the SvcFramework packages are used by the service supplier while the Framework and AppFramework packages are used by the client application.

5.2.3 Framework interfaces

Framework interfaces are indexed with "Ip" which stands for "Interface Parlay" (e.g. IpCall). To distinguish callback interfaces from these network interfaces, the acronym "App" is added after Ip at the client interfaces (e.g. IpAppCall) or "Svc" for the service supplier (e.g. IpSvcCall). The Framework is composed of 7 modules: trust & security management, event notification, integrity management, service registration, service subscription, service discovery, and service factory. Each of them are described below.

The trust and security management provides the first point of contact for a user to access the framework, the ability to register (i.e. to authenticate) to the framework and the ability to select a service or to access another framework. This module is composed of the following interfaces: IpInitial, IpAuthentication and IpAccess. The IpInitial interface is the one allowing the first access to the framework. Its reference can be retrieved via a Web page or an LDAP directory. It also allows gaining access to the IpAuthentication interface, which provides authentication mechanisms (such as the one using CORBA[1] security if CORBA is used for the implementation). Once authenticated for a session, the client application can access, thanks to the IpAccess interface, other frameworks or the services subscribed by its enterprise operator. Parlay service access encapsulates digital signature and agreement text that provide non-repudiation for both parties in agreeing that the service would be available for use.

[1] Common Object Request Broker Architecture, see http://www.omg.org.

Figure 5.2 shows the sequence diagram for an authentication. Once the client application has retrieved a reference to the IpInitial interface, it can invoke the initiateAuthentication() method to initiate the authentication process. The client provides a reference to its authentication interface (e.g. IpAppAuthentication) and receives a reference to the IpAuthentication interface. The client then uses this interface to select the authentication method. Once the method is chosen, the client and framework authenticate each other using the prescribed method. Upon successful (mutual) authentication, the client invokes requestAccess() on the IpInitial interface. The client provides a reference to its access interface (e.g. IpAppAccess interface). The framework returns a reference to its IpAccess interface. This reference can then be used to access other frameworks or access a service.

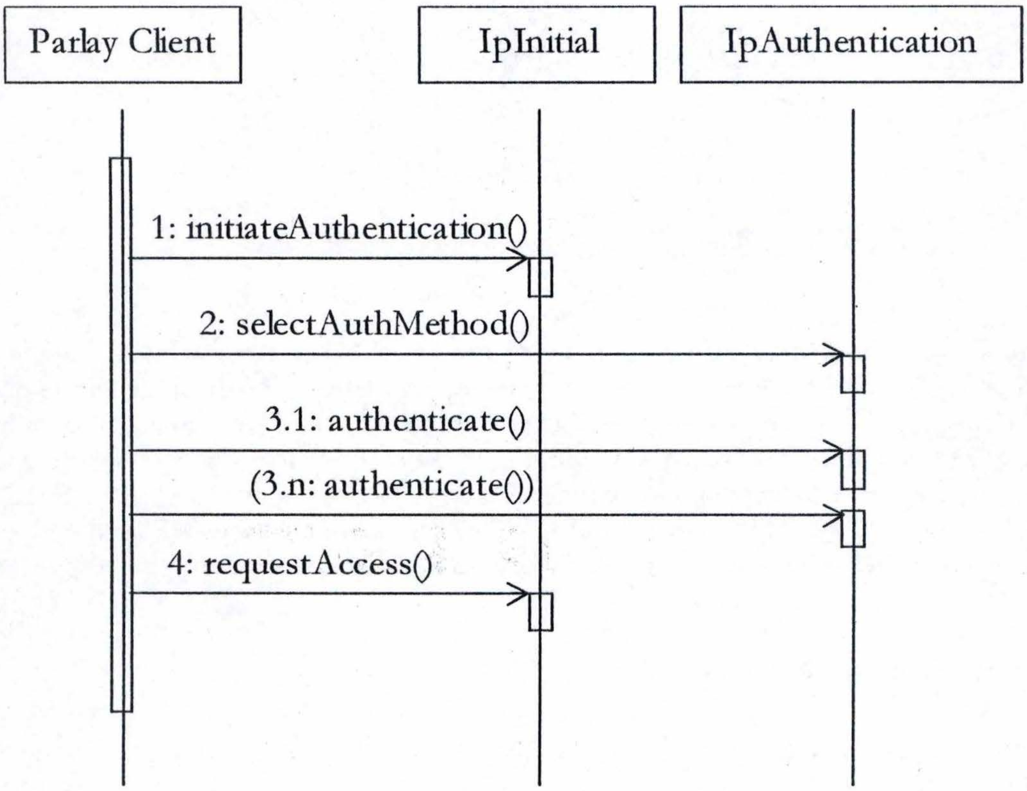


Figure 5.2 – Parlay authentication example

Figure 5.3 shows the sequence diagram for a service access. We suppose that the client application has authenticated successfully with the Parlay framework. It implies that the framework has a reference to the IpAppAccess callback interface (see requestAccess() above).

First the client must select the service (s)he wishes to use. Either the client knows the serviceId (s)he wishes to access or (s)he retrieves it by using the service discovery module. Once the service is chosen, the client invokes selectService() on the IpAccess interface. The framework returns a service token identifying this service. The client may invoke accessCheck() to check (s)he possesses enough rights to access the specified service (or a part of the service, known as a service feature). Sometimes it is important the client agrees to use the service (e.g. non-repudiation). This illustrates the use of callback interfaces. Indeed, the framework invokes signServiceAgreement() on the callback interface (IpAppAccess) to provide the service agreement text the client has to digitally sign and return. The same goes afterwards for the framework for the non-repudiation of both parties.

Once the agreement text has been signed by both parties, the client receives the service manager interface of the requested service.

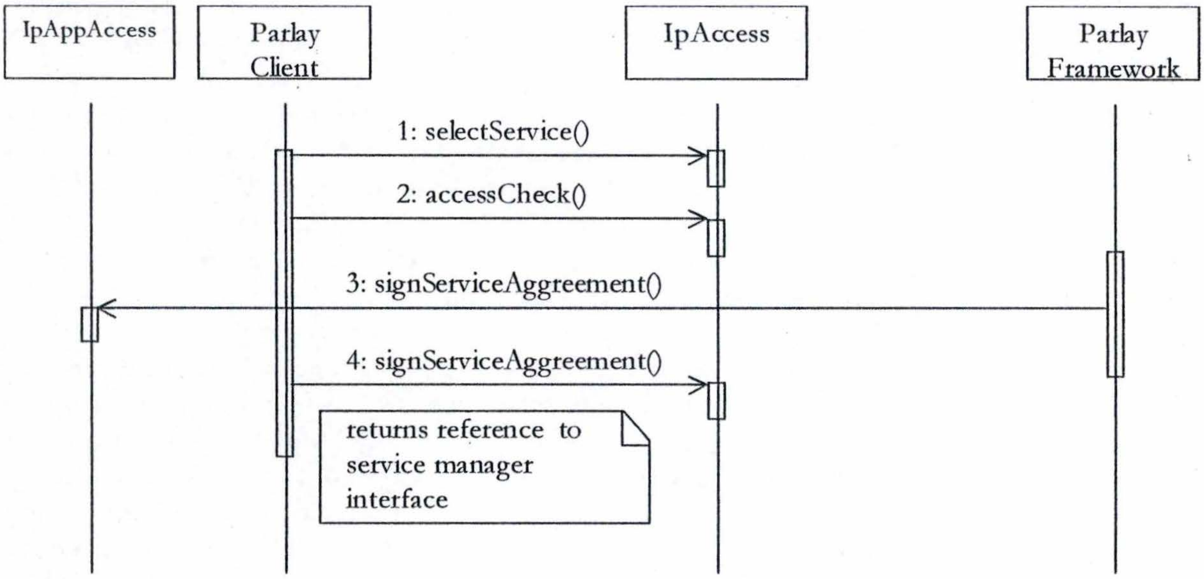

IpAppAccess
Parlay Client
IpAccess
Parlay Framework
1: selectService()
2: accessCheck()
3: signServiceAggreement()
4: signServiceAggreement()
returns reference to service manager interface

Figure 5.3 – Service access example

The event notification module is used to notify the application of generic service related events that have occurred. The IpEventNotification interface provides method to enable/disable the notification. The user must, in the traditional way, implement the callback interface (IpAppEventNotification for the client application) to allow the framework to notify him/her. Figure 5.4 shows how a client application can enable the notification for a range of events.

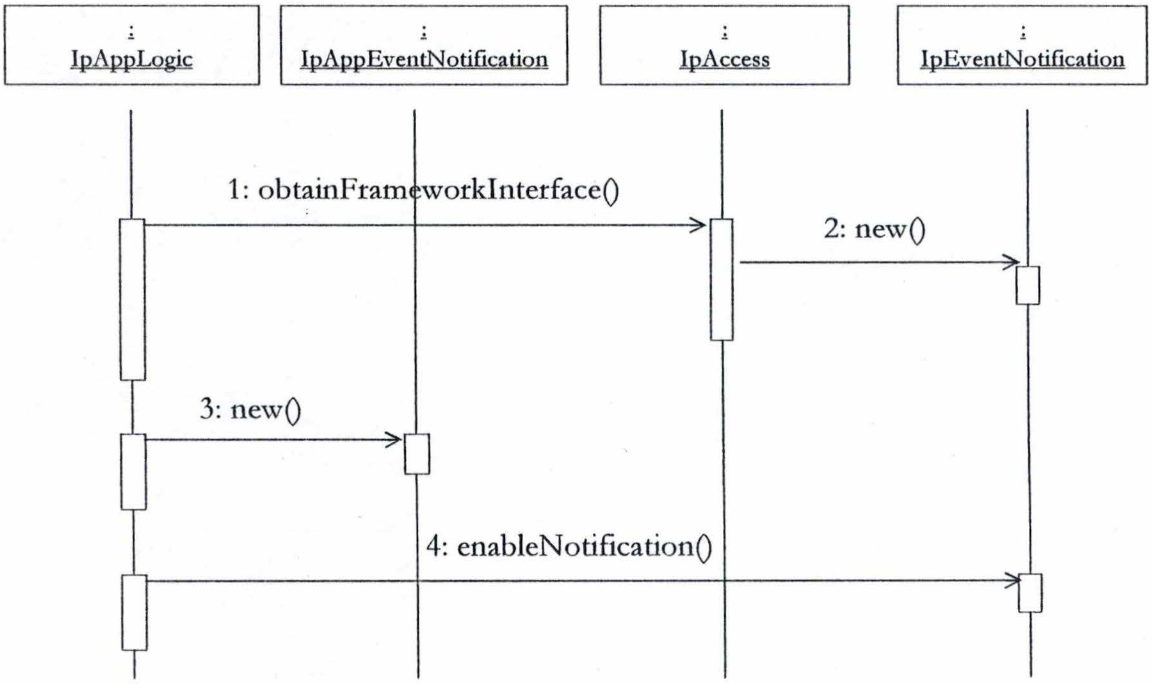


Figure 5.4 – Event notification activation example

The integrity management module is composed of the following features: load manager, fault manager, heartbeat management and Operations Administration and Management (OA&M) interfaces. The load manager features allow the load to be distributed across multiple machines and across multiple component processes, according to a load management policy. The load management policy identifies what load management rules the framework should follow for the specific client application. It might specify what action the framework should take as the congestion level changes. Briefly, the load management policy is related to the QoS level to which the application is subscribed. The fault manager is used to inform the framework about events that affect the integrity of the system, and to request information about the framework and services. The heartbeat management is used to ask for supervision in a keep alive request/answer model. The user invokes the method on this interface to ask the framework to send him/her periodical heartbeats. The fault manager uses this system whenever a service crashes, for instance. The OA&M interface is used to query the system date and time.

The service registration module allows service suppliers to register their services. Services are registered against a particular service type. Therefore, service types are created first (by the framework operator[1]), and then services corresponding to those types are accepted from the service suppliers for registration in the Parlay framework. The framework maintains a repository of service types and registered services. In order to register a new service in the framework, the service supplier must select a service type and the "property values" for the service. The service discovery module enables the service supplier to obtain a list of all the service types supported by the framework and their associated sets of service property values. This mechanism is illustrated on Figure 5.5.

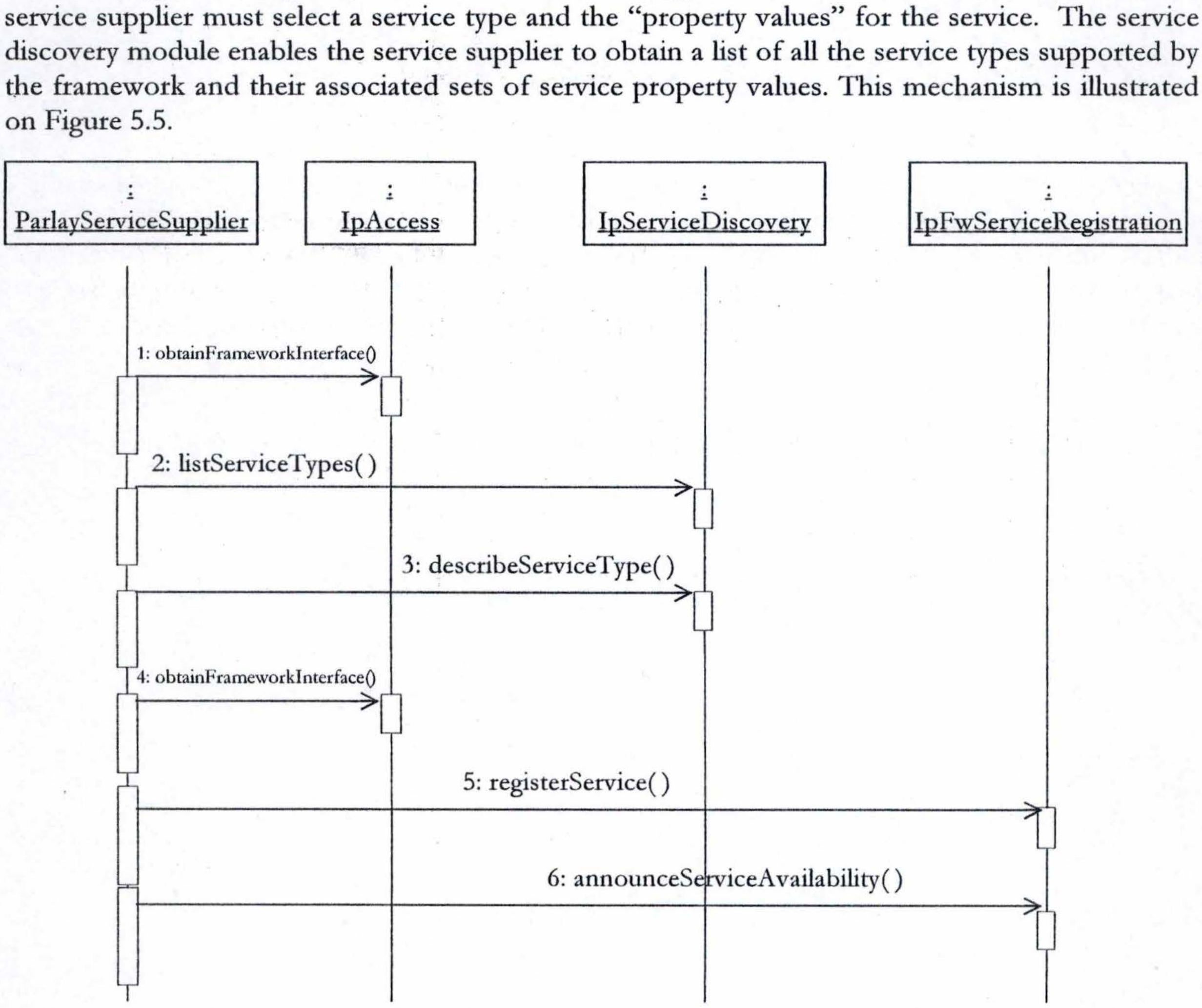

: ParlayServiceSupplier
: IpAccess
: IpServiceDiscovery
: IpFwServiceRegistration
1: obtainFrameworkInterface()
2: listServiceTypes( )
3: describeServiceType( )
4: obtainFrameworkInterface()
5: registerService( )
6: announceServiceAvailability( )

Figure 5.5 – Service registration example

[1] This operation is outside the scope of the framework interfaces.

In (1), the service supplier retrieves a reference for the service discovery module by using the obtainFrameworkInterface() method. (S)he uses, in (2), this reference to consult the service types supported by this framework. Once the service supplier has found the service type which matches the service (s)he wishes to register, (s)he invokes (3) to get a complete description of it (it is useful to know the properties that have to be filled in, as well as the range for these properties). In (4), the service supplier retrieves a reference for the service registration module in the same way as in (1). In (5), (s)he registers his or her services. The service is not accessible for the clients until the service supplier has invoked the announceServiceAvailability() for the service. Thus, the service can be subscribed after the completion of (6).

Once services are registered, an enterprise operator can subscribe to them. This subscription is performed via the service subscription module. This subscription is not always mandatory. Indeed, some services are automatically accessible to any Parlay user. The Parlay specification has its own subscription model. In this model, the enterprise operator acts as the subscriber/customer of services and the client application acts as the user or consumer of services. The Parlay framework itself acts as a retailer of services. We discuss this model in more detail in the next chapter. This module also contains management interfaces allowing the enterprise operator to create new client application accounts, service contracts, service profiles, etc. Once a contract between an enterprise operator and a service supplier has been signed, this enterprise operator can give access to the client application in its domain for this service via service profiles. The service discovery is very useful for these tasks.

The service discovery module allows the retrieval of services and service types information. As we have already stressed, it is used by the service supplier to determine which service type corresponds to the service (s)he wishes to register. This module is also used by clients, i.e. client applications and enterprise operators, to get the status of the framework in terms of services.

Each service has a service manager interface that is the initial point of contact for the Parlay service. The service factory module allows for the framework retrieving of the service manager interface of any service. Once retrieved, the framework generally forwards it to the client that requested the use of the service. This has already been discussed in Figure 5.3.

5.2.4 Service interfaces

Keep in mind that Parlay service capabilities provide applications with an abstraction of the operator's network functionalities. The services created by the service supplier represent those applications. In short, services created by the service supplier are using the Parlay services to access the network functionalities. However, Parlay services are developed like a traditional application from the framework's point of view. Each Parlay service implements a "service manager" interface which is sent to a client (here an application) requesting the use of this service. There are five Parlay services, which we describe below.

The first, Generic Call Control Service (GCCS), allows calls to be instantiated from the application and routed through the network. It supports enough functionality to allow call routing and call management for advanced services such as the ones existing in an IN. Moreover, it is the aim of GCCS to be specialized into call control specifications, such as SIP, H.323, ISUP, etc. This implies the network independence we have mentioned. The realization of this service is highly influenced by the Java Telephony API (JTAPI, see [JTAPI]) which provides these functionalities, i.e. call routing and call management for advanced telephony services. In fact, there are two ways for an application to get the control of a call. The application can create a new call from the application. Another way is to request the notification of calls that meet certain criteria, such as:

- Call initiation by an end-user
- End-user states (busy, not answering, etc.)
- Terminal states (off-hook or on-hook, for instance)

When a call occurs in the network that meets these criteria, the application is notified and can control the call. The GCCS is represented by the IpCallManager and IpCall interfaces that interface to services provided by the network. To handle responses and reports, the developer must implement IpAppCallManager and IpAppCall to provide the callback mechanism.

The second, user interaction, allows applications to interact with the end user (e.g. the client application) via predefined interfaces. This service can be decomposed into two modules. The first, the Generic User Interface Service (GUIS), is used by applications to interact with end users while the second, Call User Interaction Service (CUIS), provides functions to send information to, or gather information from the user (or call party) to which a call leg is connected. CUIS can be used to collect some digits for instance. Moreover, user interaction can be used by applications to ask the network operator to play a message to a certain call leg.

The third, messaging service, allows applications to send, store and retrieve messages (remember the Unified Messaging service of chapter 3). Current version supports voice mail and electronic mail. This service grants the application access to mailboxes, which are the main entry to the messaging system. Mailboxes are composed of folders (at least the inbox and the outbox folders). Each folder contains messages which usually have properties associated with them. Mailboxes can be opened, closed or locked and the application can request notification if certain messaging criteria are met.

The fourth, connectivity management, allows controlling QoS in the network and the last mobility, is used by wireless networks. These are not discussed further here.

5.2.5 Wrapping up

The Parlay API is an architecture which may provide network independence and application portability. This initiative constitutes an amelioration compared to the IN view. Indeed, IN services are delivered in a network centric approach. With an API-based approach, network capabilities that are useful to the applications are encapsulated and made visible to these applications. This improves the integrity, the performance and the security of the network.

However, this approach is still theoretical. For instance, the technology independence can be debated. Even if most of the Parlay implementations are CORBA-based, Parlay has not chosen a single middleware platform. This could involve lots of problems at the application level since the API cannot integrate and use the services of a specific middleware such as CORBA. Moreover, this care in the standardisation can disadvantage marginal implementations, which are using a proprietary bus to communicate between the Parlay users.

5.3 JAIN

The Java APIs for Integrated Networks (JAIN) initiative focuses on service portability, network convergence and secure network access. We discuss how JAIN is structured and we compare this approach with the results we gathered so far. Then we describe the alliance between Parlay and JAIN.

5.3.1 Introduction

The purpose of the JAIN technology is to enable the integration of the Internet and the Intelligent Network, referred as Integrated Networks. According to [JAWP01], JAIN consists of two areas of development. The first, Protocol API Specifications, specifies interfaces to wireline, wireless[1] and IP signalling protocols. The other, Application API Specifications, specifies "the APIs required for service creation within a Java framework spanning across all protocols covered by the Protocol API Specifications". The JAIN initiative is summarized in Figure 5.6.

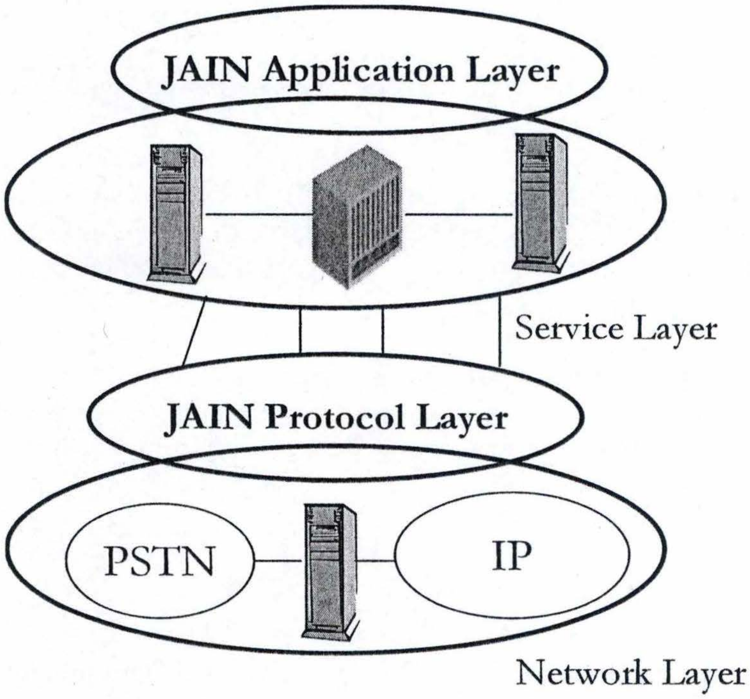


Figure 5.6 – The JAIN initiative

As shown in the figure above, JAIN proposes standardization at two different layers. The network layer implies network convergence while the application layer implies service portability. The secure network access is fulfilled by enabling applications residing outside the network to directly access network resources and devices. This corresponds to the alliance between JAIN and Parlay.

[1] Our discussion is based on a subset of the JAIN initiative since wireless networks are outside the scope of this dissertation.

5.3.2 JAIN Protocol API Specifications

This area is managed by the JAIN Protocol Expert Group (PEG). This group standardizes interfaces to IP and SS7 network mainly. This includes, for instance, TCAP, INAP, ISUP for the SS7-side and SIP, H.248 (or MEGACO) and H.323 for the IP-side.

More precisely, "the Java SS7 API defines a set of software components that allow an application to invoke a rich set of network resources while hiding many of the complexities that differentiate the SS7 variants" [LUC00a, p 21]. Moreover, JAIN supports higher-level interfaces that abstract the differences between protocols and thus ease the convergence between the PSTN and the Internet, for instance. In practice, the SS7 API allows any JAIN entity to access and communicate with an IN. Furthermore, the use of interfaces allows a network operator to implement these interfaces with her or his particular network. The ultimate goal is, of course, to completely abstract the underlying network layer.

Each JAIN SS7 API defines three major Java software components: Stack, Provider and Listener. The Stack component abstracts the underlying SS7 protocol stack. To do so, it provides a "factory" to create and manage the Provider component. Provider component maps the generic API to a specific stack (see above). Listener components interact with the Provider ones via Event objects. In order to exchange such objects, Listener components must register with Provider components.

The JAIN IP APIs allow providing services over an IP network equivalent to those of a traditional telephone network. The evolution towards NGNs can be progressive since network entities preserve their existing native protocol software while simultaneously enabling Java applications to provide new services.

Thus, JAIN-compliant entities could support and participate in the development of the NGN. Indeed, by providing standardized protocol interfaces, applications and protocol stacks can be dynamically interchanged. It also implies that such entities can use protocol stacks from different vendors.

5.3.3 JAIN Application API Specifications

This area is managed by the Application Expert Group (AEG). This group defines the Java Call Control (JCC) API, the Java Coordination and Transaction (JCAT) API, the Java Service Logic Execution Environment (JSLEE) API, JAIN connectivity management, JAIN SPA (Parlay) and, finally, JAIN service creation. The very first goal of this group is to provide a single call (or session) model across all supported protocols in the protocol layer we have just discussed above.

The purpose of JCC/JCAT is to hide underlying protocols, such as SS7 ISUP or SIP for instance, from the service programmer. The JAIN SLEE is performing just the same as a traditional SLEE (a more general SCP, as we have already stressed in chapter 2). Trusted services are hosted within JSLEE. Finally, JAIN service creation is used exactly in the same way as the Service Creation Environment Point (SCEP) in an IN.

JAIN supports three basic abstractions, as shown in Figure 5.7. At the lowest level, applications can talk directly to the JAIN adapters, i.e. the APIs defined by the PEG. The advantage is that these applications can have a fine-grained control and thus probably performance improvements. Unfortunately, this level has also some drawbacks. The development is much more complex, and the loss of the abstraction provided by the next levels could be a problem, too. For instance, an application which has to handle both the PSTN-side and the IP-side, must handle both SIP and ISUP.

In the next level, the call control can be defined independent of the underlying signalling protocol interfacing with JCC/JCAT. If we go back to the previous example, the application does not have to handle any protocol. The developer has just to use the abstraction offered by this level. Moreover, a number of additional capabilities are offered to trusted services.

The third and last level of abstraction is related to untrusted services, and thus, to the JAIN SPA interfaces which are implementing the Parlay specification. At this level, only a subset of network capabilities is made available to untrusted services such as those developed by third parties. The reason for this is, of course, that untrusted users must not have the same set of privileges as a trusted user.

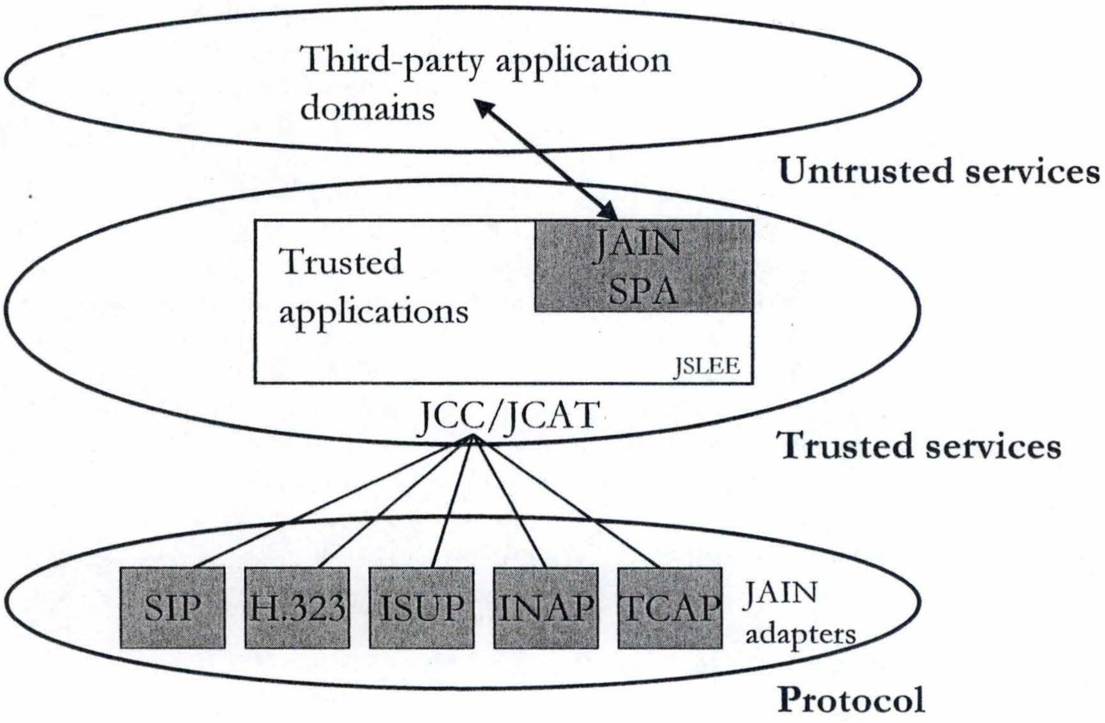


Figure 5.7 – JAIN abstractions

5.3.4 The JAIN component model

A component model is "a set of principles that define how solutions can be built from smaller (software) entities. In Java, these smaller entities are Beans, and examples of currently available beans are JavaBeans, [...]. The principles cover many aspects of the entire lifecycle [...], such as how individual beans can be assembled into larger entities and ultimately into complete solutions." [KTG00, p. 98].

The JAIN technology is based on Java component software (i.e. JavaBeans). Components can be added, enhanced, assembled, shared, or redistributed in a dynamic running system. This allows services and features to be added, updated, and deleted in a live environment.

The JAIN component model has learned well from the Intelligent Network environment. Indeed, the properties and features of JavaBeans in a JAIN system are clearly much better than the static SIBs that are used to create traditional IN services. After IN's promise that has never been respected, this is the first time that a complete and open architecture would support components reuse.

5.3.5 JAIN Parlay alliance

Since June 2000 (see [SUN00]), JAIN and Parlay have joined forces. The result of this alliance consists of an API that translates the Parlay specification that we have described above: the JAIN Service Provider Access (SPA) API. This API allows any JAIN-compliant entity accessing a Parlay framework and, thus, access services that are supported by this framework.

This alliance was not straightforward since these two initiatives use mechanisms that are neither supported, nor best suited in the Java language. [BGD00] describes these problems and their associated solutions. We only describe the bigger difference since the other is much more performance-oriented.

Parlay uses asynchronous methods and callbacks while JAIN uses the JavaBeans event model to fulfil the same role. The JAIN Parlay Edit Group attempted to incorporate the JavaBeans event model into JAIN SPA. However, the callback model was chosen for two key reasons. The first is that event listeners register with only a single event source. It implies that events could not be filtered, which would lead to unwanted events being sent to the event listeners. Even if the listeners per event source could be limited to one, the events would still not be filtered, which would still lead to unwanted events being sent to the event listener. For this reason, the callback event model was chosen since it allows filtering events to be sent to a single callback object. The second is that asynchronous methods are mostly used in real-time telecommunications systems because of their non-blocking nature. Even if this approach has not been used within the JAIN Communitiy, this makes it ideal in situations when applications are multitasking calls.

5.4 Web call center example

In this example, we discuss how the notions we have covered could be illustrated in the example we have relied on since chapter 3: a web call center. As you may recall, there are two ways to access a service from the softswitch. The first is to contact an IN and uses the traditional process. The second, that we discuss here, is to use an applications server.

Remember the hypotheses of our example: a client, whatever protocol it is using, accesses a softswitch to contact a web call center at number +32-800-12345. This softswitch is, of course for our example, JAIN-compliant. It means that it hosts a JSLEE implementation and thus the JAIN SPA interfaces. Our applications server implements a Parlay framework. Finally, somewhere in the network, the web call center is ready[1] and is Parlay-compliant. These hypotheses are schematised in Figure 5.8.

[1] "Ready" means that it has registered with the framework and thus the latter knows about it.

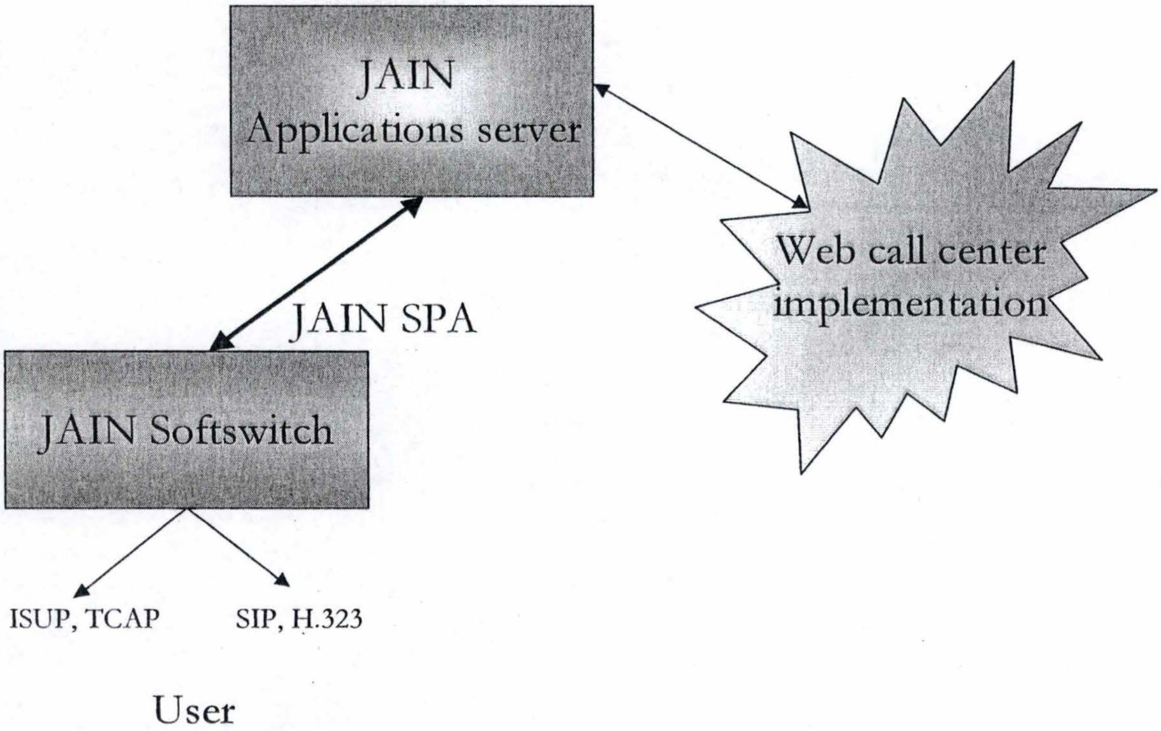


Figure 5.8 – JAIN Parlay entities

As shown in Figure 5.8, the network to which the user belongs is completely abstracted thanks to the JAIN adapters. First of all, the user has to contact the call center via a freephone number. Upon reception of the call attempt, the internal policy of the softswitch says that it must forward this call to the application server. Now the application server, i.e. the Parlay framework, is responsible for that call. In the remainder of this example, we discuss the needed services that are activated to allow the access to the call center. Our discussion is based on sequence diagrams describing message flows between the actors.

The differences between JAIN Parlay and the other examples we have covered previously is that the first manages everything, even number translation. Our illustration for this example is thus subdivided into 2 sections. The first one is devoted to number translation while the second one shows the call center functioning.

5.4.1 Number translation

Number translation is eased by the GCCS service. Generally it would be developed internally, i.e. by the framework operator. But in large systems, it could be done by application developers themselves in order to serve a set of services based on freephone number access. We suppose that this service is developed internally. The sequence diagram for our number translation service is shown on Figure 5.9.

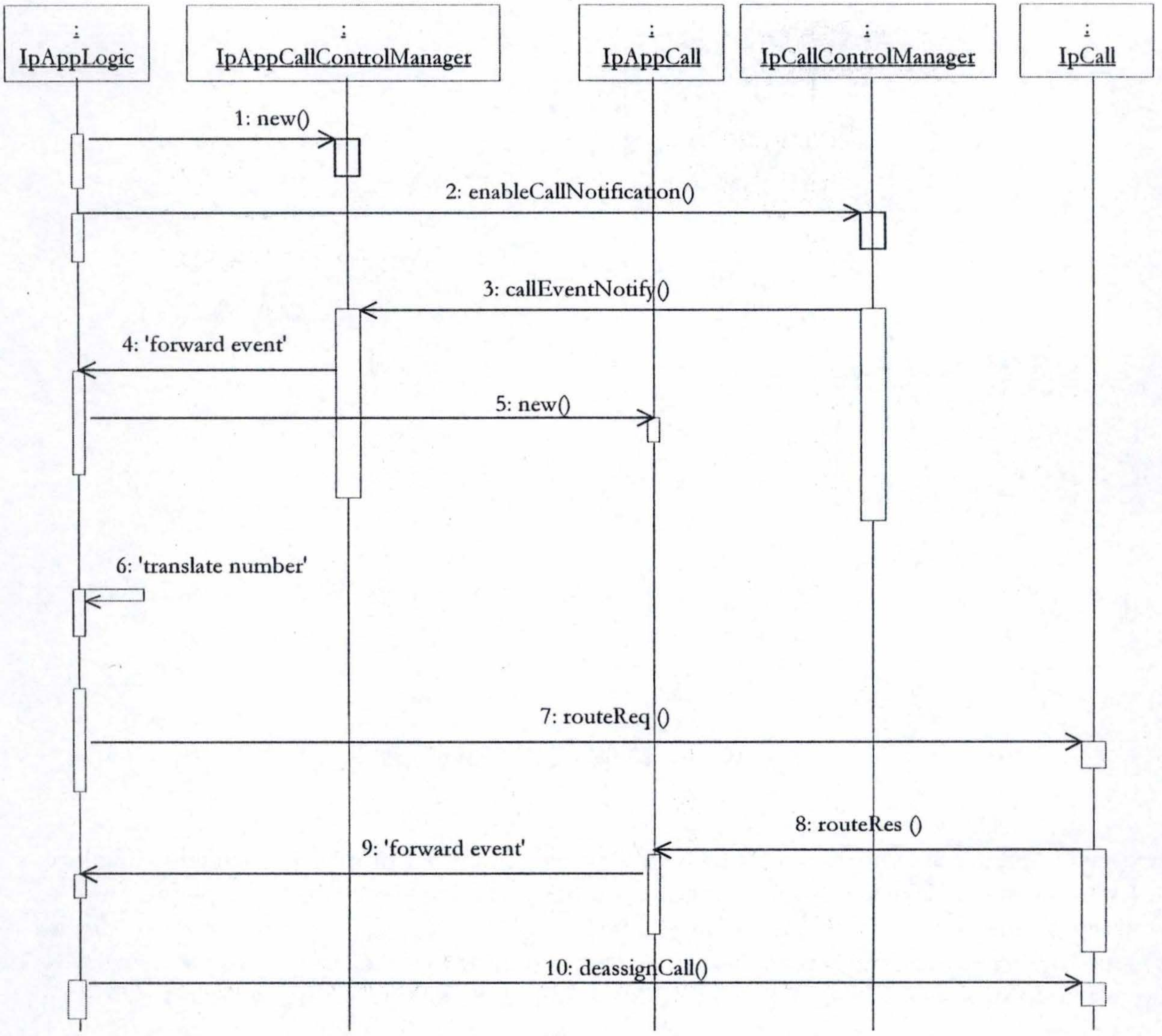


Figure 5.9 – Number translation sequence diagram

In (1), the application creates an object implementing the IpAppCallControlManager interface. This object provides the application call control management functions to the GCCS. In (2), the application enables notifications on new call events. In our restricted example, the event is related to the +32-800-12345 number but it is more likely that such a service manages an address range rather than a single number.

These operations are performed before our call takes place (again this represents the fact that the service is "Ready"). Suppose now that a call arrives and matches the criteria set in (2), the IpCallControlManager notifies the application that have requested it, in (3). In (4), the event is forwarded to the IpAppLogic.

Our goal is to route the call to the call center. To do so, the service creates, in (5), an object that implements the IpAppCall interface, which will be used for callbacks. In (6), the service performs the number translation function. The result of this function is used to route the call towards the destination (7). We suppose that there is no complication here for the called party (not answering, busy, etc.). In (8), the result of the call being answered is returned to the callback object, which notifies the AppLogic in (9). Since the call has been routed successfully, the application is no longer interested in controlling the call and therefore deassigns the call (10). The call will continue in the network, but there will be no further communication between the call object and the application.

This first service shows how the Service interfaces can ease the development of a service. Everything in this application is already provided by the Parlay system. The only thing the application has to do is translate effectively the number (step 6). This also shows how the development time can be dramatically reduced.

Now the call has been properly routed, we explain what happens when the call center service interacts with the client. When talking about clients, do not forget that the Parlay client here is the softswitch.

5.4.2 Call Center

In fact, the number translation process can be incorporated into the call center application. Since we have covered a generic number translation service in the previous section, we integrate our web call center in it, i.e. we discuss in more details step 6 of Figure 5.9. Upon receipt of the event, the application can retrieve some information about the calling party (and here we are talking about the user and not the softswitch). The call center can route the call only based on this information but it could be useful to get more information from the calling party before routing the call. We illustrate this in Figure 5.10 by showing how a prompt and collect method can be used.

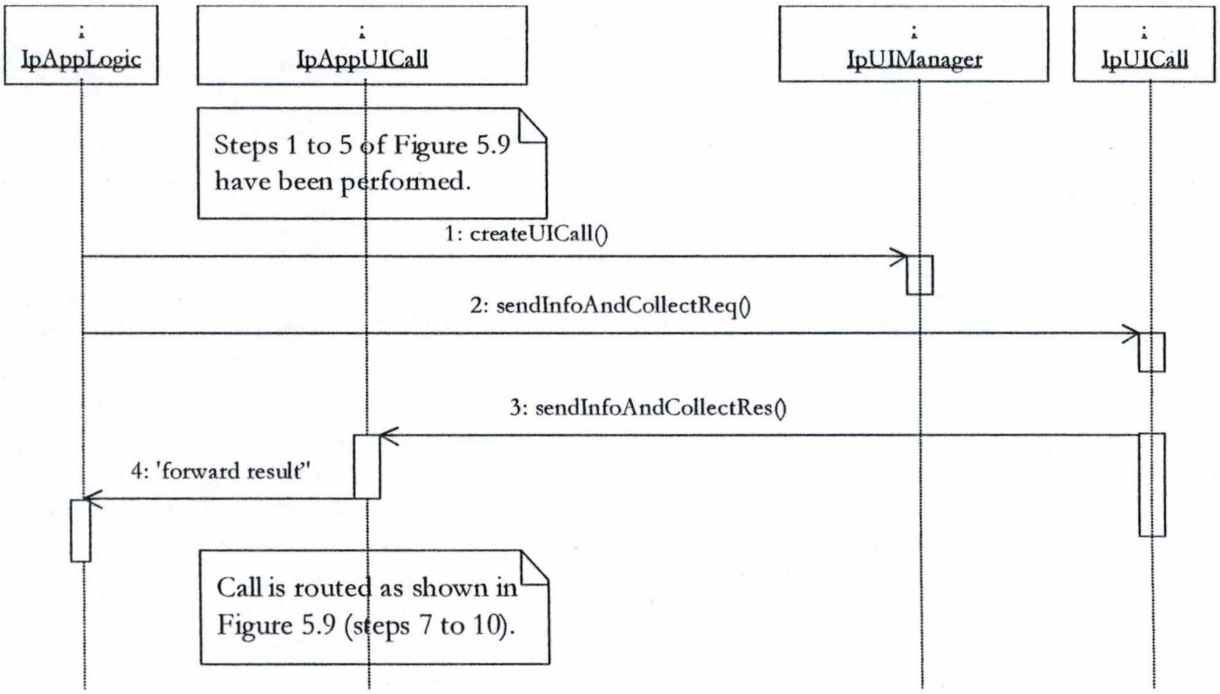

: IpAppLogic
: IpAppUICall
: IpUIManager
: IpUICall
Steps 1 to 5 of Figure 5.9 have been performed.
1: createUICall()
2: sendInfoAndCollectReq()
3: sendInfoAndCollectRes()
4: 'forward result"
Call is routed as shown in Figure 5.9 (steps 7 to 10).

Figure 5.10 – Prompt and collect sequence diagram

The service could request the softswitch to connect the calling party with special equipment playing a pre-recorded voice: "enter 1 for American cottage, enter 2 for Canadian cottage, …". In an IN, this equipment is known as an SRP (Service Resource Point). To do so, the application can use the Call User Interaction Service (CUIS) that we have already discussed.

We are supposing that the application has received an event related to our a call center and is ready for number translation. In (1), the application associates a user interaction object with the calling party. The initial service dialogue is invoked using this message (2). The information to be sent is specified by using an ID. Of course the application does not know about the equipment in the network and thus is not able to address them directly. In fact, the Parlay framework manages everything: the application has only to know the ID to which the pre-recorded voice belongs. The result of the dialogue, which in this case is the number of the request department code (e.g. American cottage or Canadian cottage), is returned to its callback object using (3) and forwarded via (4) to the IpAppLogic. Once the application has this extra information, it can route the call to the requested department (this is already illustrated in Figure 5.9, steps 7 to 10).

The fact that the Parlay client is a softswitch is not a problem. Indeed, it initiates the request by giving the calling party information that it has received when the user instantiates the call to +32-800-12345. The digits collection is also not a problem. Once it receives this request, it can order the MG to connect to the specified equipment. Once the user has entered the information code, the softswitch routes this result through a JAIN SPA interface.

One could think that this implementation is less interesting than the SIP one. This is actually the opposite. In SIP, the current information (e.g. the page that the user is surfing to) could be used by the CGI to route the call. Thus, this is completely transparent to the client. In JAIN Parlay, the prompt and collect method is adapted to traditional phone. When talking about a web environment, i.e. the user is surfing, (s)he gets a pop-up window giving her or him the choices. This could also be done transparently if the softswitch knows the structure of the service (i.e. which code belongs to which department).

5.5 Traditional IN vs. JAIN Parlay

Across this dissertation, the initiatives we have covered refer to the Intelligent Network. Some of these, e.g. PINT and SPIRITS, are trying to make IN interworking more or less with the Internet. Indeed, their view is to add specific equipments to traditional one's in the aim of being able to interwork with packet-based networks, such as the Internet. We have also briefly touched the SIP world with SIP CGI and sipplet. All these initiatives do not represent, in our opinion, a system that is able to evolve, based on the services' requirements. Indeed, PINT is strongly limited to the milestone services the user can access. This is not especially blameworthy and we do not find this choice wrong anyway. But this significantly limits the long-term view, the support of all (IN) services across all networks. We even support such initiatives since they will probably bring lots of experience for the future.

The other view we have covered, JAIN Parlay, focuses more on the long-term view. This choice implies that no implementation is available at the present time. JAIN Parlay brings lots of new concepts. As a conclusion, it would be interesting to compare this technology with IN. This is what we are going to do now.

5.5.1 Traditional IN: a closed system

Even if the specifications present IN as an open platform, able to interwork with other SS7 networks, IN has always been a vertically closed system. Because strategic choices were different, lots of SS7 networks are truly different and offer weak interworking. Then, the network that we have designed as the IN platform is proprietary by nature. It belongs to one and only one provider. Most of the time, this provider also develops IN services that are provided through their platform. In this proprietary environment, it is not straightforward to interwork IN platforms from different vendors.

IN has been designed during a time where the Internet had not such a great importance as today. It has also been designed without the experience that lots of specialists have today. IN, and the traditional network in general, are too closed. Their architecture could not be modified, as the NGN states. It implies that the telecommunication network of today does not offer a good interworking with the Internet.

We have seen that it is possible to offer services to Internet users. For instance, the reload of a prepaid card from a web site. Once the request has been received by the web server, it can contact an IN platform to process the request actions. Actually, this weak interworking remains a problem since the provider has to develop two different services. One for the traditional network and one for the Internet (that will most probably uses the first one but with specific additions related to the Internet).

The increasingly importance of the Internet and the problems with data flows in telecommunication networks has called for a change. Service providers are aware of this need and are working towards new ways of providing additional and innovating services.

5.5.2 JAIN Parlay: the solution?

The JAIN Parlay system is an open based, programmable system. Compared to IN, it offers lots of improvements and a total interoperability between different networks. In chapter 5, the call center example has shown how services can rely on basic features to fasten the service development. Is JAIN Parlay the solution for the shortcomings described above?

The JAIN technology offers an abstraction at three levels. The first offers network abstraction. Since the system is completely programmable, when a vendor writes a stack for his equipment, all systems can be upgraded in order to be able to communicate with it. This constitutes a major improvement compared to IN: this feature offers interoperability across networks and a total compatibility with traditional switches, designed by different vendors. Of course, to do so vendors have to write the stack related to their equipments. The stack refers to the JAIN adapter and some industrials have begun to release JAIN adapters for their equipment.

The second level provides a unified call model for the application. Application developers can use this level to write applications that can run on IP, the PSTN, wireless networks, etc. Thanks to the JAIN adapters, this feature is completely transparent to the application developer. It involves that a single service is able to offer access through various networks (e.g. the reload of the prepaid card needs only one service).

In traditional IN, services are composed of a chain of Service Independent building Blocks (SIBs). Most of the time, IN services need common facilities (e.g. connect a leg to a special equipment). Nobody will deny that SIBs are well suited for the traditional network's environment. Again, with the emergence of the Internet, they come to their limits. SIBs are difficult to share or update. In fact, they have been designed for specific needs and offer, most of the time, a weak flexibility. JAIN works with the same model but has learned from the SIB's experience. In JAIN, components are represented by JavaBeans. Such beans can be shared, updated and distributed in a live environment. They offer the flexibility of Object Oriented (OO) programming as well as the inheritance between beans. To our opinion, the Java language has a powerful network stack (java.net and java.io packages) and ease the development of distributed applications.

Finally, the last level of abstraction offers a feature that has never existed in the traditional network: third-party development. Parlay offers a framework which manages the support and security of such development. Untrusted parties can rely on some of the features of the network to develop and provide their own services. Moreover, the JavaBeans model implies that users can share components and can even constructs APIs to improve the flexibility and fasten the service creation.

In fact, the JAIN Parlay technology could be the platform for supporting IN services in the NGN. Lots of work has to be done but we believe that it is worth to do them.

5.6 Conclusion

Chapters 4 and 5 have shown how the new generation of services can be programmed and supported. Chapter 4 provides hybrid solutions with PINT, SPIRITS and SIP GCI, even if the last one gets away from hybrid networks and is closer to NGNs. In this chapter, we have shown how services can be supported in open, programmable networks like NGNs.

One could find that there are too many solutions for this specific problem but, fortunately, each solution addresses different problems. Firstay, PINT & SPIRITS can provide rich services that use both the Internet and the PSTN. Once the architectural limitations enounced in chapter 3 are resolved, SIP (or another signalling protocol) could provide flexible and really interesting services, such as our web call center. The integration with the Internet is stronger here. Indeed, the operator can send pages to the client in this case while with PINT & SPIRITS this is not possible.

Finally, the Open API network is the last step towards NGNs in terms of services. A lot of things still have to be done, especially concerning the JAIN adapters as well as JAIN-compliant hardware and software.

Chapter 6

A JAIN Parlay (pre) implementation

This chapter resumes a JAIN Parlay pre-implementation using a commercial IN platform from Alcatel. This chapter is structured in 4 parts. First of all, we describe the environment in which this implementation has been performed. Then, we introduce briefly the pieces of the Parlay specification that have been implemented. We go on to the specification of the Distributed Processing Environment (DPE) router and its stack implementation. We then conclude this chapter by describing the Parlay implementation that has been performed using this stack and the issues we encountered.

6.1 Implementation environment

The Alcatel platform, also known as an Open Service Platform (OSP), allows the introduction of IN services by supplying all common parts to the developer. Indeed, the developments are eased thanks to usage of a graphical tool chain taking the development into account from specification to integration in the customer environment. Practically, the OSP is composed of a multitude of services: each of them provides one or more service(s) to the others.

The purpose of this implementation is to add a service which has the same functionalities as the Parlay framework we have already discussed in the previous chapter. The pre-implementation term has been chosen since our work is more an analysis of the specification as well as a test implementation than a really final work. One of the purposes of this work was also to determine whether it was a good idea to implement such a framework as an IN service or not. This pre-implementation has focused on pieces of the standardisation, mainly the framework. These are described in section 6.2.

The choice of implementing the Parlay framework as a traditional IN service is not without consequences. The Service Creation Environment Point (SCEP) is well suited to develop traditional services. For a prepaid card for instance, the service collects the digits entered by the user and analyses them. Once this is done, it connects the user with the requested number by using a special billing scheme. These tasks can be implemented very efficiently with some SIBs. However, things are very different when talking about Parlay. SIBs which are used in the PSTN are clearly not suited to allow the development of such a framework.

The majority of existing Parlay implementations today utilizes a middleware (e.g. CORBA) to retrieve and use the Parlay interfaces. At Alcatel, the (F)-DPE[1] (Fast Distributed Processing Environment) router is used by the OSP to exchange objects between nodes. The DPE is an asynchronous bus allowing the management of messages between services. It addresses these messages based on various parameters such as the requested function, the current machines' load and availability. Since the DPE is strongly integrated into the OSP, it was logical to use it instead of another middleware.

[1] In the remainder of this chapter, the term DPE is used to designate the term F-DPE as well.

The communication between Parlay users and the Parlay framework is performed thanks to the DPE. Since our approach is to use the Java language at the client-side, it is mandatory to create a DPE stack, written in Java, allowing an easy access to the OSP. The implementation of this stack was the most difficult task: it resulted from an in-depth comprehension of the protocol that is used over the DPE as well as a specification of all the events that can occur while using it. This stack is described in section 6.3. The reader is invited to consult the "Dpe Stack Report" document for more implementation detail.

The fact that this implementation has not been performed from scratch is its main characteristic. On the one hand, it can rely on the DPE and the SIBs which are used in traditional services. On the other hand, it has to tackle with the limitation of this environment. After having described the way this test implementation has been performed (section 6.4), we go on with the issues we encountered (section 6.5). We end this chapter by drawing our conclusions (section 6.6).

6.2 Parlay

As you may recall, the JAIN initiative has released a set of interfaces that translates the Parlay specification into the Java language. These interfaces form the JAIN Service Access Provider (SPA) API. In this chapter, we linger over version 2.1. This version provides mainly a translation of the framework part with the jain.spa.framework package. The JAIN SPA interfaces are used by the client to communicate with the framework. This client uses intensively the DPE stack to abstract the specificities of this bus. The development of this client is outside the scope of our implementation even if we have developed a small one for test purposes.

We have also seen in chapter 5 that the framework is composed of 4 APIs: Framework and AppFramework for the client and FWFramework and SvcFramework for the service provider. In our implementation we aggregate the FWFramework and Framework packages in only one package: an IN service residing on the platform[1]. The two others packages are used for callback purposes, as we have also seen in chapter 5.

The following functionalities have been implemented in this test version. The framework implements:

- Trust and Security management
- Service discovery
- Service subscription
- Service registration

These modules are described in more detail below. The Parlay specification offers additional functionalities that have not been implemented. These are the integrity management and event notification modules. The reason is that the platform already supports such services and they do not have to be implemented. However, the existing mechanism should be modified, if needed, in order to be Parlay-compliant.

[1] In the remainder of this chapter, the term framework is used to designate this IN service.

6.2.1 Trust and Security management

The trust and security management interfaces provide:

- The first point of contact for a user to access the framework
- The ability to register (i.e. to authenticate) to the framework
- The ability to select a service or to access another framework

The ability to access another framework has not been implemented here, since the platform together with the DPE allow access to other platforms as well. By first point of contact, the Parlay specification means a way to initiate a contact between a user and the framework. This could be done via LDAP for example (this remains a detail, so we are not discussing it further). In short, our implementation allows registering to the framework and selecting a service that the user can access.

6.2.2 Service discovery

All Parlay actors use service discovery. Enterprise operators use it to know the services they can offer to their clients. Clients use it to know which services are available. Service providers use it to know which services are available on this specific framework. They can add afterwards a service that is not already available. Our implementation provides all the features proposed in this module and is thus the one which achieves the most.

6.2.3 Service Subscription

The service subscription module includes a lot of functionalities, mainly focused on management. In this module, the enterprise operators can manage their clients, services, service contracts, etc. The Parlay specification has its own subscription model (see [PASP00a, p. 58]):

"Services may be provided to the Enterprise Operator directly by a service provider or indirectly through a retailer, such as the Parlay Framework. An enterprise operator represents an organisation or a company which has a number of users associated with their client applications. Before a service can be used by the client applications in the enterprise operator's organisation, subscription to the service must take place. An enterprise operator subscribes to a service by (electronically) signing a contract about the service usage with the Parlay Framework, using an on-line subscription interface provided by the Framework. The Framework provides the service according to the service contract. The Enterprise Operator authorises the client application in his/her domain for the service usage. Finally a subscribed service can be used by a particular client application."

"The enterprise operator provides to the Parlay Framework the data about the client applications in its domain and the type of services each of them should be allowed access to, using the subscription interfaces offered by the Framework. The Framework provides (to the enterprise operator) the subscription interfaces for subscriber, client application and service contract management. This gives the enterprise operators the capability to dynamically create, modify and delete, in the framework domain, the client applications and service contracts belonging to its domain."

6.2.4 Service registration

Before a service can be brokered (discovered, subscribed, accessed, etc.) by an enterprise, it has to be registered within the Framework. Services are registered against a particular service type. The framework maintains a repository of service types and registered services accessible via the service discovery module. In order to register a new service in the framework, the service supplier must select a service type and the "property values" for the service. Then the service supplier uses the functionalities of the service registration module to effectively register his or her service. For instance, assume that the service type "internal" represents all the services that can be settled in the platform itself. Assume also that a client must give an address, a Service Access Point (SAP) and an IN object to instantiate the service. To represent this, the service type could have, for instance, three service type properties. The first one defines the range of IP addresses, the second the pattern of the SAPs and the last a String one (an id for the needed object). If a service supplier whishes to register an internal service, then (s)he is obliged to provide this information in the form of 3 service properties (for instance, 138.48.32.107, "cliapp.create", "ParlayClient"). Our implementation allows the service supplier to register a service and is in conformance with the specification.

6.2.5 Parlay services

The Parlay specification is composed of two parts. The first is the framework we have just covered. The other one is the service part which provides a set of common service features. These can be used by service developers to fasten their implementation. Clearly, this part was not in the scope of this implementation. Our approach is mainly "client-oriented". Thus, the implementation provides all mechanisms to manage and access the framework. The service part (and finally the use of the services) is not covered by our implementation.

6.3 The DPE Router

The Distributed Processing Environment (DPE) router is a proprietary software bus that allows real-time communication between platform services and IN services. The DPE can also communicate with other external nodes through an IP network. This very functionality is attractive for our task, since Parlay clients can use Java applets or applications located on the customer's PC.

The DPE allows a client that possesses some rights to access the platform with the aim of executing a service (for example translate a freephone number depending on all the factors we have seen so far). Thus, once the framework is implemented, Parlay users can use the DPE to access the functionalities we have covered above. In the remainder of this section, we discuss in detail the implementation of the Java stack that allows the access to the DPE.

6.3.1 Types of DPE messages

Like any other bus, the DPE router has its own syntax in terms of message which we call from now on DpeMessage. The generic DpeMessage can be decomposed into information messages and IN messages. Information messages are lower-level information. IN messages are decomposed into object messages and string messages. The first describes an IN object that the platform can recognize (e.g. an instance of a prepaid card). The latter allows simply transporting an array of characters (String). The message structure is summarized in Figure 6.1

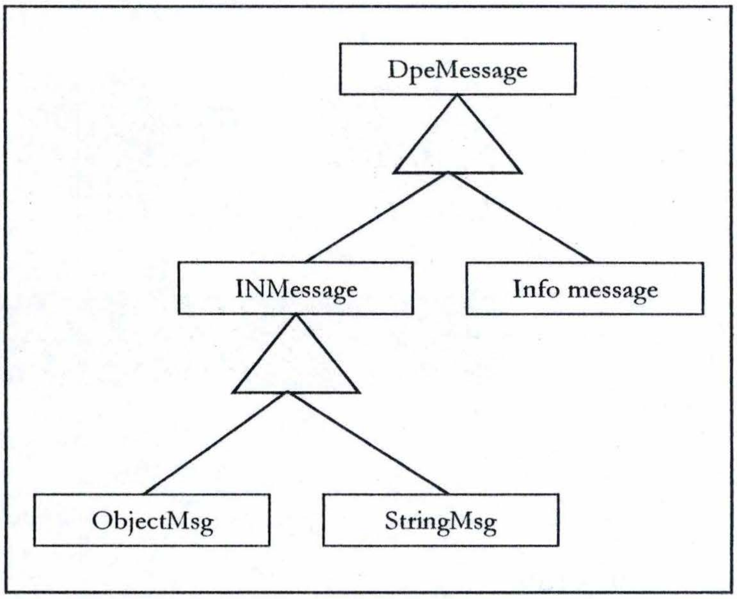


Figure 6.1 – DpeMessage hierarchy

Generally, developers use only IN messages while information messages are used by system administrators. IN messages are useful to instantiate a service or a functionality with parameters.

6.3.2 DPE interface stack functions

The purpose of the DPE access stack is to free the developer from the complex syntax of DpeMessage so we do not discuss this further.

The stack must:

- Allow the developers to send DpeMessage only based on as little information as possible (e.g. IP address and SAP for the needed service).
- Serialize and unserialize messages to and from the platform
- Events handling
- Free the developers with dialog establishment and handling (watch-dog mechanism, etc.)

6.3.3 Communication infrastructure

The DPE library allows a Java client to connect to the DPE router through an IP network. Then the SAP contained in the message allows the DPE to route the message to the proper service. Roughly, this composed the needed information to send a message to a service. The DPE library and the DPE router are performing the remainder on the fly. Figure 6.2 shows how a Java client can access the Alcatel platform in the aim of performing some tasks.

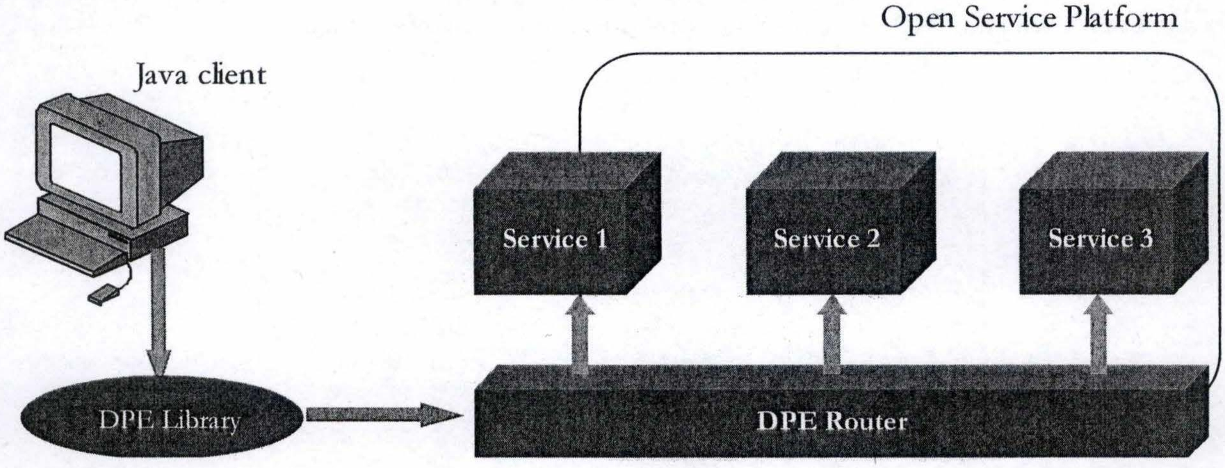
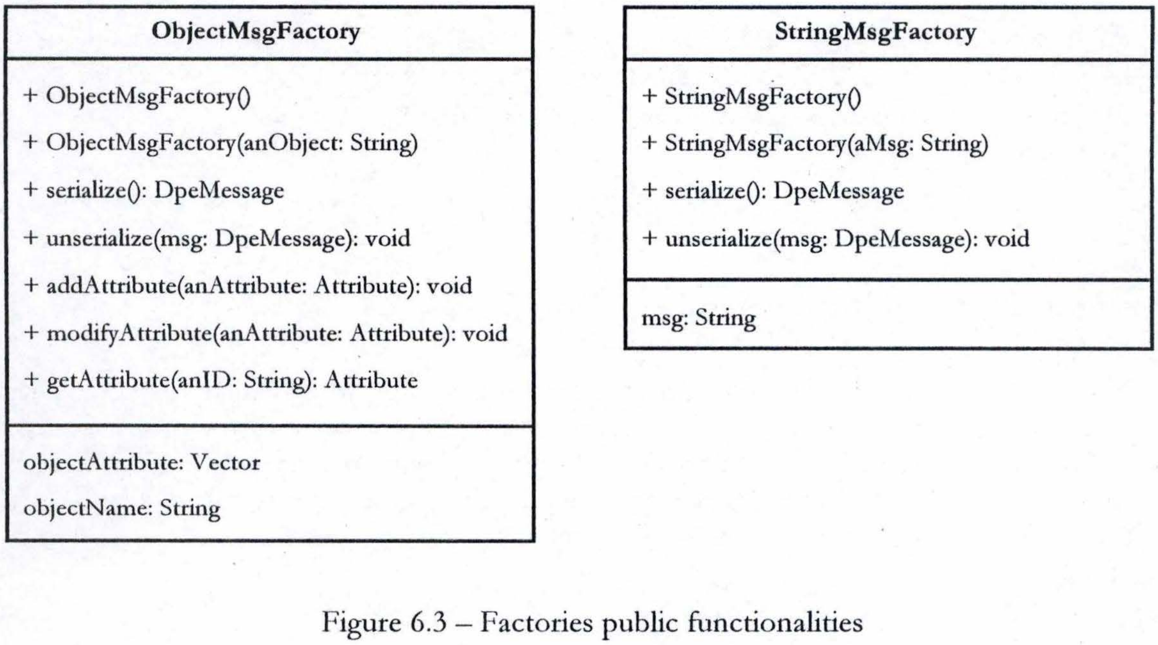


Figure 6.2 – Communication infrastructure

6.3.4 Message serialization

Suppose that a Parlay client – which is an enterprise operator – wishes to create a new client. The enterprise operator has to fill in the needed information in a "client" object. The way this object is transmitted at the other end, i.e. the Parlay framework, is not a relevant information. Message serialization allows here to transform a Java object into a DpeMessage. Unserialization allows converting back a DpeMessage, coming from the server, into its Java representation.

Since the message structure and the protocol used by the DPE is relatively complicated, it is important to allow the developer to create DpeMessage automatically. The factories fulfil this role. As shown in figure 6.1, IN messages are composed of object messages and string messages. The DPE library contains two very important objects: ObjectMsgFactory and StringMsgFactory. Figure 6.3 summarizes their public functionalities.



ObjectMsgFactory

+ ObjectMsgFactory()

+ ObjectMsgFactory(anObject: String)

+ serialize(): DpeMessage

+ unserialize(msg: DpeMessage): void

+ addAttribute(anAttribute: Attribute): void

+ modifyAttribute(anAttribute: Attribute): void

+ getAttribute(anID: String): Attribute

objectAttribute: Vector

objectName: String

StringMsgFactory

+ StringMsgFactory()

+ StringMsgFactory(aMsg: String)

+ serialize(): DpeMessage

+ unserialize(msg: DpeMessage): void

msg: String

Figure 6.3 – Factories public functionalities

An object message is characterized by a name and a list of attributes. A string message is only characterized by the string itself. A factory represents, at a given time, the status of an IN object. For example, an instance of StringMsgFactory can represents the string message "hello, world!". If the developer then uses the serialize() method, (s)he recovers a DpeMessage that represents this message.

6.3.5 Events handling

There are two types of events related to the DPE library. You may recall that the Parlay specification uses intensively the callback mechanism. Thus the Parlay server must have the ability to send asynchronous messages to registered clients. Message notification is the first type. The other one is basic information notification.

In Java, there is a very powerful mechanism used to perform this task: listeners. In Java, an event or a group of events are related to a listener. If a developer wants to be notified about these events, (s)he has to implement the interface related to these events and add it to the object that is generating them.. The same applies for DPE events: each type of event has its own listeners. DpeMessageListener for asynchronous message management and DpeEventListener for DPE event management. Figure 6.4 shows the listeners declarations.

```
public interface DpeMessageListener extends EventListener
{
        /**
         * Invoked when a DpeMessage arrives.
         */
        public void DpeMessageReceived(DpeMessageEvent e);
}
public interface DpeEventListener extends EventListener
{
        /**
         * Invoked when an event occurs in the system.
         */
        public void DpeEventReceived(DpeEvent e);
}
```

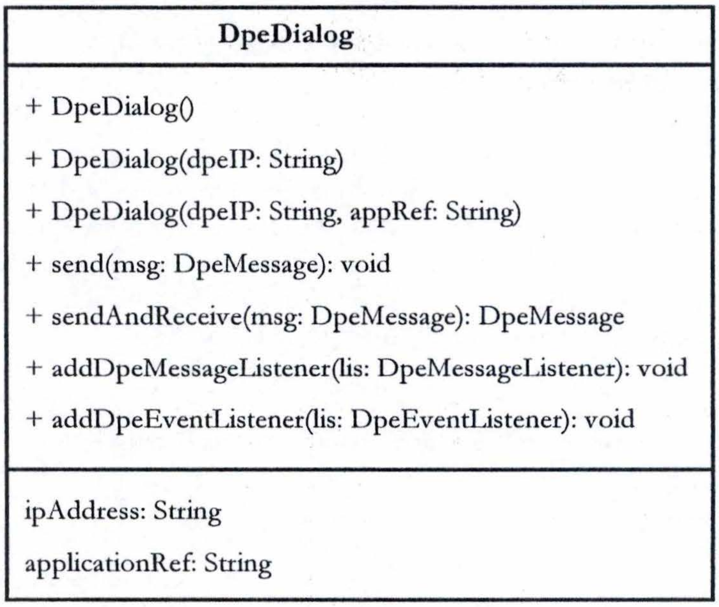
Figure 6.4 – Listeners declaration

6.3.6 Dialog establishment and handling

Dialog establishment and handling is the ultimate aspect to automate. Indeed, the developer does not have to care about the flags contained in DpeMessages, the way they are transmitted and received, etc. A central object in the DPE Library is the DpeDialog object. In our context, a dialog can be defined as an association between a client and an IN service. When a DpeDialog is instantiated, it launches a number of threads which are waiting for incoming messages or manage the link by using the watch-dog mechanism. Figure 6.5 shows the public functionalities of this object.

The DPE library allows sending synchronous and asynchronous message. This results in two methods: send() and sendAndReceive(). The first is invoked to send an asynchronous message and perhaps retrieve asynchronous response afterwards thanks to the listener. The latter is used to retrieve directly the response.

The DpeDialog is also the object that is generating DPE events. Thus it offers two methods to allow the developer to register the interfaces we have seen above. When a thread receives an asynchronous message, the DpeDialog instance invokes the DpeMessageReceived() method for all the interfaces that have been registered with this dialog.



DpeDialog

+ DpeDialog()
+ DpeDialog(dpeIP: String)
+ DpeDialog(dpeIP: String, appRef: String)
+ send(msg: DpeMessage): void
+ sendAndReceive(msg: DpeMessage): DpeMessage
+ addDpeMessageListener(lis: DpeMessageListener): void
+ addDpeEventListener(lis: DpeEventListener): void

ipAddress: String
applicationRef: String

Figure 6.5 – DpeDialog public functionalities

We conclude this part with an example of service invocation. The mechanism we describe here is schematised in Figure 6.6 below.

Suppose that a Parlay user wants to know which services are available. To do so, (s)he has to contact the Parlay framework: in our case, this is an IN service located in the OSP and accessible through the DPE. This user utilizes the DPE library we have seen so far. First the factory translates the request into a Java object representing a DpeMessage and asks a dialog establishment between itself and a well-known DPE router. This results in an invocation of a DpeDialog instance. This instance launches automatically a thread (DpeDialogReceiver) which handles incoming messages from the platform.

Then, the client sends its request by invoking the send() (or sendAndReceive) method. Upon reception of this message, the service performs the requested action. Once this is done, it sends back a DpeMessage containing the response of the user invocation. Some requests are requiring more than one message to be sent and this could take a while. The DpeDialogReceiver handles the dialog by using the watch-dog mechanism. If a problem occurs in the network, the user that has registered for DPE events is notified as soon as possible.

Once the DpeDialogReceiver gets the response from the platform, it notifies internally its DpeDialog instance. Then the latter checks if the request was synchronous or asynchronous. If it is synchronous, it sends the message directly to the user by ending the blocking sendAndReceive() method. In the other case, it sends a notification by using the listener we have seen previously.

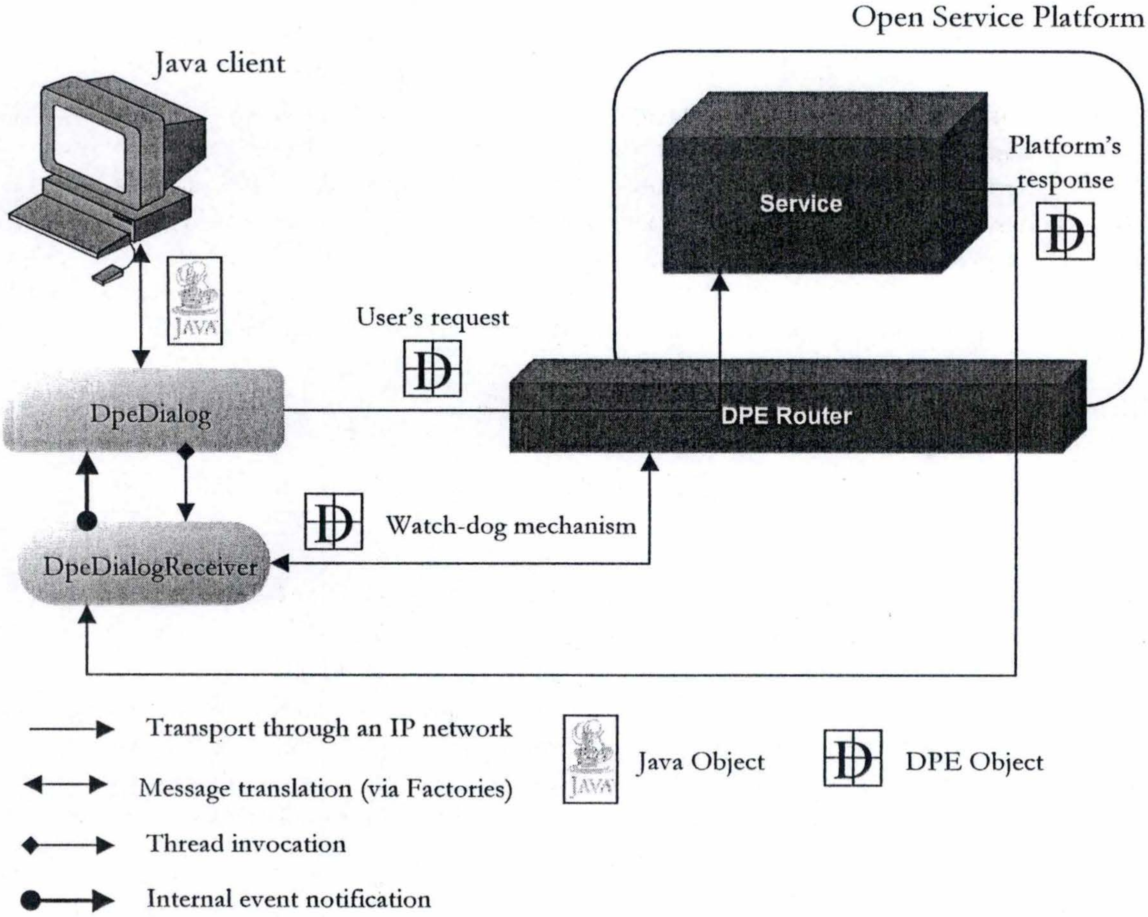


Figure 6.6 – Feature invocation example

6.3.7 Evaluation and conclusion

The Java stack we have presented here allows contacting the OSP and its functionalities. It allows synchronous and asynchronous requests and responses. It handles low-level functionalities. It offers listeners mechanism which allows the user to be notified for classes of events.

The current version is suited for our pre-implementation. However, a solution based on CORBA, for instance, is better suited for a Parlay framework that has to respond to a lot of users. Even if the DPE is a powerful software bus, it has not the standardisation of CORBA. It implies that it would be very difficult to interconnect a Parlay framework that is using the DPE router with any other solution based on CORBA.

6.4 A JAIN Parlay (pre) implementation

This section depicts the implementation of the modules described in section 6.2 with the help of the DPE router we have just covered. These modules are implemented the same way any other

IN service is implemented. The JAIN SPA interfaces are used by the users to access and use the Parlay framework. In other words, they implement the callback interfaces for the specified user.

6.4.1 Introduction

Parlay defines interfaces. For each interface, Parlay defines different methods. Our framework supports one connector for each method of interface. A connector is a subset of the SAP used to address the requested service. For example, the connector "cliapp.create" is associated to the createCliApp() method of the cliapp objet. The JAIN SPA interfaces, implemented in clients, use these connectors to request the needed action.

The implementation uses a variety of existing SIBs that are used to:

- Send and receive DpeMessage (from/to the client)[1]
- Access a database
- Perform checks related to the specification (i.e. checking the rights of a Parlay user)
- Authenticate a user

The framework works in a request/response model. As we have already seen, the DPE library translates the JAIN SPA object into a DpeMessage. This DpeMessage is then translated into an IN message that the platform can understand. This message contained the SAP for the needed functionality (i.e. the "address" of the service and the specific connector to instantiate this service). Once the request has been performed, the framework sends back a DpeMessage, which is translated into a JAIN SPA object. The Parlay specification provides a number of exceptions that can occur as well as the possible answer and their formats. The results that come from the framework are described in section 6.4.3.

6.4.2 Framework database

Since our work must be considered as a pre-implementation, we have not represented all the possible attributes of the Parlay actors, as defined in the Parlay specification. The framework database scheme is summarized in Figure 6.7 below.

The pentop object represents an enterprise operator, which is described by an id and a password. The pcliapp object represents a client application. Each client application belongs to one and one enterprise operator. The psersup is structured the same way as the pentop one, but to represent a service supplier this time.

The pservice, pserprop, psertype and pstprop objects represent a service, a service property, a service type and a service type property, respectively.

The pcontra object represents a service contract between a service supplier and an enterprise operator. It specifies also additional information, such as the date to which the contract is valid (start date) and invalid (end date).

[1] In the implementation environment, this SIB is called SROBJ which stands for Send and Receive OBJects. SROBJ is illustrated in section 6.4.

The psag and psagmem objects represent a Subscription Assignment Group (SAG) and its members respectively. A SAG is a group of clients which can access the same set of services.

Such objects are critical for the functionalities we have implemented. Indeed, the enterprise operator identification is critical for the trust and security management module, for instance. Service types are used by the service provider to register his or her service in a well-defined class of service. Services information is used by the service discovery. Service contracts are used to check the rights associated to a service, etc.

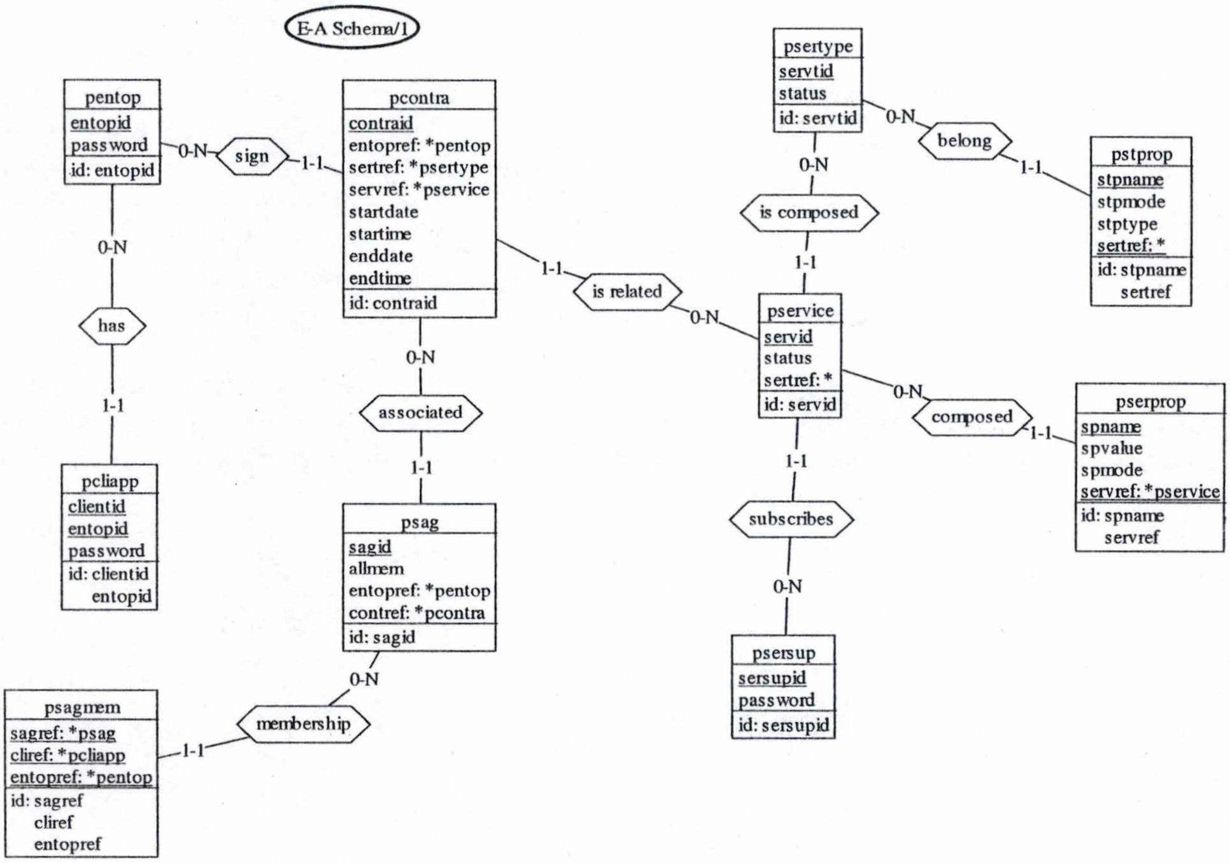


Figure 6.7 – Framework database scheme

One could think that this decomposition is really bad. In fact, the environment in which the implementation has to take place has limited our decomposition. This is also the reason for the ugly object's names: IN object's names are limited to 8 characters maximum.

6.4.3 Framework's responses

The framework's responses are decomposed into two groups. First, IN objects, which are the Java representations of objects contained in the platform database. Typically, this is a services list as a result of a service discovery request. The other group is composed of 3-digit integer result codes. They are subdivided in the same way as HTTP or SIP responses are:

- 1xx: Informational – query is processing
- 2xx: Success – the request has been performed successfully

- 3xx: Not used[1]
- 4xx: Client or service error
- 5xx: Framework error
- 6xx: Global error

In result codes of class 2xx, the second digit represents the corresponding user. For instance, a 211 code means that the client application account has been created successfully. A 222 code means that the specified enterprise operator account has been successfully modified, etc.

6.4.4 A functionality example

The purpose of this chapter is not to cover all functionalities in detail, so only one is given: client application account creation. The purpose of this request is to add a client to the specified enterprise operator domain. Once this client has been registered to the parlay framework with a user name and a password, (s)he can use the functionalities that his or her enterprise operator has allowed him or her to use (see the Parlay subscription model described in section 6.2.3).

Figure 6.8 shows the UML diagram describing this functionality. Once the enterprise operator has authenticated with the framework, (s)he uses the IpAccess interface to obtain an IpCliAppManagement interface. Indeed, this interface contains the createClientApp() method which is then invoked.

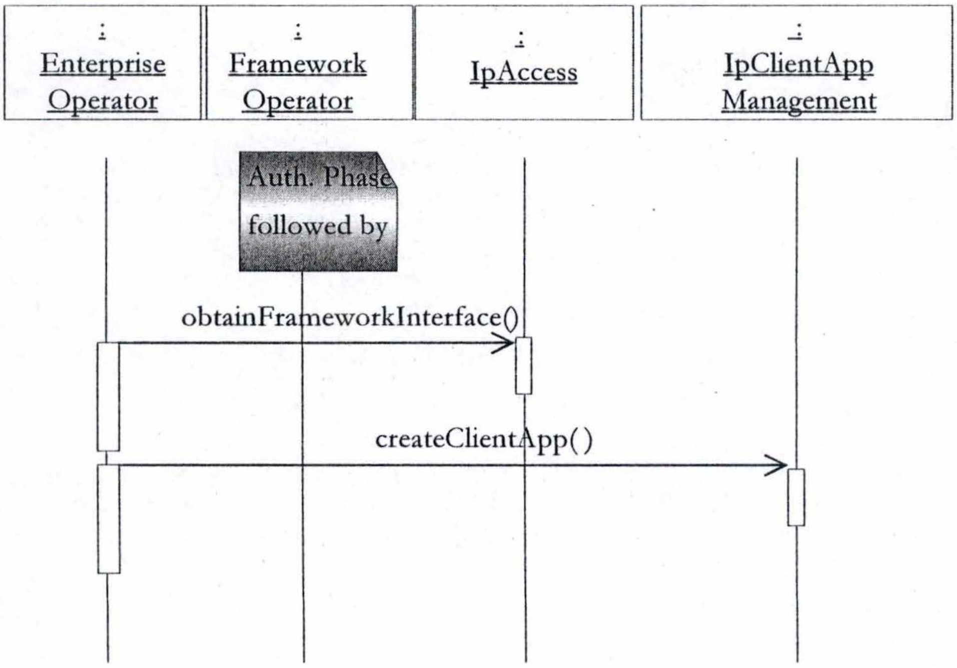


Figure 6.8 – Client application account creation sequence diagram

Every functionality has been specified in a pattern. This pattern is subdivided into four planes. The first briefly describes the purpose of the method. The second specifies the message(s) that has(have) to be sent. The third gives the side effect(s) on the database. The last enounces the

[1] Clearly, this class can be used when the implementation shall consider a multi-framework environment.

possible result codes and the reasons of their invocations. We provide in Table 6.1 the specification of the createClientApp() method as an example.

cliapp.create	IpClientAppManagement.createClientApp()	
Used to create a client application account. A specific reference to the enterprise operator is automatically added to the account.		
SROBJ	Message expected	Timeout
1	The enterprise operator information stored in a **pentop** object	-
2	The client application account to create	15
Side Effects on the database		
A **pcliapp** object is created with the client application information stored in the message received in [2].		
Messages returned to the Client		
211	CLIAPP_ACCOUNT_CREATED if the operation is successful	
403	USER_BAD_PASSWORD if the password provided in [1] is not correct	
404	NOT_FOUND if the user is unknown to the framework	
407	UNEXPECTED_DATA if the SROBJ do not receive the expected object	
416	CLIAPP_ALREADY_EXIST (the user must use cliapp.modify to succeed)	
500	DATA_RECEIVE_ERROR if the SROBJ cannot decode the object	
501	DATA_SEND_ERROR if the SROBJ cannot send a message (+ alarm)	
521	DB_CREATION_ERROR if the account creation is not successful.	
602	CONNECTION_TIMEOUT (if the 15 sec timeout expired)	

Table 6.1 – A functionality specification

Two messages must be sent. First, the enterprise operator identification, which is checked by the framework (this is represented by the "authentication phase" square in Figure 6.8). Second, the client application that has to be created. The side effect on the database is the creation of a new tuple in the pcliapp table. Finally all events that can occur are covered and are related to a specific result code as well.

6.5 Implementation issues

Our objectives were to analyse completely the Parlay specification and to propose a test implementation. Moreover, the creation of the DPE stack was also a mandatory step in our process. The Parlay specification is rather important so we focused only on the framework at the client-side. The results of this work are composed of documents explaining the framework specification and its implementation, the DPE stack code, its documentation and a short tutorial.

This pre-implementation was not straightforward and we are even puzzled about it. The use of the DPE instead of a more standardized bus like CORBA could be surprising at a glance. However, this solution is logical since our implementation has been injected into an OSP which is exclusively using the DPE to exchange objects.

The request/response model of the system has brought some issues. At first, it was not planned to implement the specification in this model. It implies that the authentication module is not designed elegantly. Indeed, the user has to authenticate each time (s)he wish to use a functionality of the framework. This model also increases the global efficiency and thus the response time. Moreover, it is critical to have a connection-oriented model to provide the integrity and event management modules. The platform uses other mechanisms to provide this but it is clear that the support of these modules could be a headache.

The problem of using traditional SIBs has already been touched shortly. The main limitation of this approach is its inflexibility. It should be noted that we could have the possibility of creating our own SIBs and then strongly increase the overall flexibility. Unfortunately, we do not have enough time – and skill in this proprietary environment – to consider this solution as feasible. As a consequence, too much of the logic has been delegated to the client, written in Java and providing much more flexibility for this environment. This delegation has important security problems: since the framework works as a servant, anyone writing its own client can do whatever (s)he want.

6.6 Conclusion

The integration of a module into an existing system has brought us a dual view of our work. On the one hand, the existing system brings a lot of standard functionalities such a development tools, debugging facilities (e.g. traces), etc. On the other hand, these functionalities can, in a way, limit the possibilities.

The development of the DPE stack is our greatest pride. The Alcatel platform together with Java clients can bring new possibilities. This stack is not limited to the purpose of Parlay. Indeed, it can be used by a servlet to allow users on a web site to access a traditional service, such as the reload of a prepaid card for instance.

We hope that the analysis and the development of the Parlay specification could bring enough experience to fasten and ease the development of a Parlay framework from scratch. In our opinion and based on the experience we have in the OSP, it is worth nothing to try to desperately integrate new system with old ones. Indeed, we have shown across this chapter that the OSP is not best suited to integrate efficiently a Parlay framework.

However, professionals are, most of the time, improving solutions in place of creating them from scratch. Clearly, the OSP has lots of potential and could most probably be extended to efficiently support the new paradigm of the PSTN-IP interworking.

Chapter 7

Conclusion

This report presented some of the different initiatives allowing the delivery of a new generation of services. We first presented what currently exists and also showed how it could evolve to interact with the Internet. Then, we studied how it was possible to take profit from this architecture to extend communication among users more and more. This report was not designed as a well-determined approach to a well specific protocol meant for the professional who wishes a particular solution. The subject is, indeed, too recent for us to tackle or even to choose a particular protocol. All initiatives deserve our deepest attention since each of them brings its assets in terms of experience and originality.

7.1 Wrapping Up

Chapter 2 showed us that intelligent networks did not reach the objectives they had been designed for: development speed, independence and above all total interoperability. Two reasons were raised: firstly, the divergent view some providers can have about IN resulting from an incompatibility between the networks. The latter is consolidated by the second reason, i.e. using proprietary material. Configuration problems - linked to the too limited telephone terminal- were also debated. However, this chapter has shown to what extent IN is a success in the telephone network. Services, such as the freephone service, bring a huge amount of revenues throughout the world. It would be thus interesting to either extend IN or have it interacted with the Internet.

Technical aspects were studied in chapter 3 allowing to bring the Internet nearer to the telephony network. Indeed, it is important to study and master these techniques in order to obtain an interesting and scalable system. Techniques and inherent issues of both networks were put forward. Firstly, QoS is a critical aspect for the efficient support of voice over IP. At the same time, other techniques, such as the gateway location or the address translation from IP addresses to telephone numbers and vice versa, are crucial because they allow to rub out some of the differences between the networks. The objective of this presentation was not -once again- to try to be exhaustive but on the contrary to give a maximum of directions to someone wishing to deepen a particular detail later on. Finally, we suggested a generic model allowing the delivery of services on such a network.

The last three chapters presented the initiatives allowing an efficient use of this support. Chapter 4 offered the so-called current solutions in the way that they could be -and are in some parts of the world- applied for immediate use. One could think, for instance, of the Click-to-Call service allowing a user to ask a call establishment between himself / herself and a company. These intermediates solutions are using both networks (i.e. the traditional one and the Internet one) and do not require the immediate development discussed in Chapter 3. Indeed, the Click-to-Call service establishes the call in the traditional network. It can be extended to the support of voice over IP. Once the support of VoIP will be effective, such a service should evolve very easily. Solutions based on SIP are longer-termed since they require the support of VoIP, which is not straightforward, especially in public networks.

Chapter 5 lays out slightly advanced solutions -more than probably based on the outcomes delivered by the solutions set out in the preceding chapter. The JAIN Parlay technology resolves the current problem (i.e. network convergence) but it also supports the NGN paradigm since the technology is completely independent of the underlying network. Thus, JAIN Parlay can work with an SS7 network or an IP network transparently. Of course, this technology does not rub out the aforementioned restrictions discussed, the ones of Chapter 1. It means that all JAIN Parlay-compliant services are not available from a simple telephone.

Finally, chapter 6 constitutes the practical point of view of this report by describing an analysis and an implementation of testing the technology and presented in the previous chapter. The particularity of this work was its environment. Indeed, the purpose was to implement the JAIN Parlay specification in a classical IN platform.

Our contribution meets two major axes. Firstly, an effort in order to integrate different initiatives emerging in this field has been achieved. This integration has been moulded around a key-example, which allows the reader to compare the strengths and weaknesses of both systems more easily. Secondly, the implementation of the Parlay framework into an existing platform underlined the inadequacy of intelligent networks with the Internet in their current design more strongly.

Since its birth, the telephony network has been working in a rather heavy and vertically integrated model. That model -as we demonstrated in this dissertation- was not designed for the model in use in the Internet today. The latter allows the speedy development of services into an integrated environment. As a matter of fact, it appears clearly to us that the Internet architects have been able to take profit from the experience relating to the telephony network, even if basically the use is not the same.

7.2. Future of Intelligent Networks ?

This issue is worth being debated. The advent of the vocal feature under IP has intrinsically and automatically suppressed particular services very much in use in the telephony network. The "follow-me" service is such an example as it allows the user to be contacted in different areas on the basis of one and only one phone number. This mode is automatically administered in VoIP since the user only needs to record himself each time he changes to a different area.

A key-issue is still to be approached, leaving aside NGNs which we find too conceptual for our conclusion: "do Intelligent Networks still have a future in an Internet-based world ?". The IPv6 support would allow every user to get a unique address with a permanent identification. The increase both of the technology and the band width - either the wireline or the wireless versions- would allow the delivery of telephony services with an even better reception quality than the one currently prevailing in the classical network. Freedom of movement, portability, would be -in this environment- meaningful concepts. Although our argumentation does not give much chance to intelligent networks, we must point out that they will only disappear if the traditional network disappears first. Considering the investments for the past few years, it is an utopia to bet on its disappearance.

In a short-term view, many efforts will be made in order to deliver services using both networks. The call center -used as a central example in this report- seems to us to be a very good illustration. The invests allocated should -as usual- slow down the start up of an NGN-type model. Market will determine the way how the paradigm touched upon in this report will evolve.

Appendix 1

Glossary

Advanced Intelligent Network (AIN)

See Intelligent Network (IN). AIN is rather an American concept. In Europe, IN refers to the same concept.

Application Programming Interface (API)

Means by which a programmer writing an application program can make requests of another application, an underlying operating system, or a communication infrastructure.

Asynchronous Transfer Mode (ATM)

Connection-oriented multiplexing and switching method utilizing fixed-length 53 bits cells. It is asynchronous in the send that cells carrying use data need not be periodic.

Basic Call State Model (BCSM)

A high-level finite state machine model of call processing for basic call control (i.e. a two party non-IN call). The model might only cover a portion of a call attempt, e.g. an originating BCSM or terminating BCSM, or the whole attempted call connection.

Call Control Agent Function (CCAF)

A functional entity that provides network access functions for users, interacting with call control functional entities (see CCF) in providing services.

Call Control Function (CCF)

Refers to call and connection handling in the classical sense (i.e. that of an exchange). The CCF is the Call Control (CC) function in the network that provides call/service processing and control.

Callback mechanism

Way to provide asynchronous calls by providing a callback interface at the invocation step.

Codec

Compressor/decompressor. Software component which translates video or audio between its uncompressed form and the compressed form in which it is stored. Common codecs include those for converting analog video signals into compressed video files or analog sound signals into digitised sound.

Common Channel Signalling System number 7 (SS7)

Global standard for telecommunications defined by the ITU-T. It defines the procedures and protocol by which network elements in the Public Switch Telephone Network (PSTN) exchange information over a digital signalling network to effect wireless and wireline call setup, routing and control.

Common Gateway Interface (CGI)

Interface to the Web server that enables programmers to extend the server's functionality. CGI enables one to extend the capability of the server to parse (interpret) input from the browser and return information based on user input. On a practical level, CGI is an interface that enables the programmer to write programs that can easily communicate with the server.

Common Object Request Broker Architecture (CORBA)

Architecture that enables pieces of programs, called objects, to communicate with one another regardless of what programming language they were written in or what operating system they are running on.

Customer Premise Equipment (CPE)

Wide range of customer-premises terminating equipment which is connected to the local telecommunications network. This includes telephones, modems, terminals, routers, set-top boxes, etc.

Distributed Processing Environment (DPE)

Also known as Fast Distributed Processing Environment. The DPE is an asynchronous proprietary software bus that allows real-time communication between platform services and IN services. The DPE can also communicate with external nodes through an IP network.

Dual Tone Multi Frequency (DTMF)

System used by touch-tone telephones. DTMF assigns a specific frequency, or tone, to each key so that it can easily be identified by a microprocessor.

E.164 address

A number from ITU-T Recommendation E.164 numbering plan which uniquely indicates a public network termination point. This address utilizes up to a maximum of 15 digits

Frequency Division Multiplexing (FDM)

Multiplexing technique that uses different frequencies to combine multiple streams of data for transmission over a communications medium. FDM assigns a discrete carrier frequency to each data stream and then combines many modulated carrier frequencies for transmission.

Gatekeeper

H.323 entity that provides call control services, such as address translation and, optionally, bandwidth management to other H.323 entities.

Gateway

Combination of hardware and software that links two different types of networks.

Global Title Translation (GTT)

Universal type of addressing supported by SCCP. The user can identify the destination of the message by providing a translate number (e.g. a directory number). GTT is responsible for the interpretation of this number.

Hybrid network

Network involving two kinds of network (e.g. IP and the PSTN).

In-band signalling

A format for digital signalling where the 8 bits per channel of signalling information is transmitted within the 64 kbit frame along with the data being transmitted. This creates an effective bandwidth of 56 kbps

Integrated Networks

Integration of the Internet and the Intelligent Network.

Integrated Services Digital Network (ISDN)

International communications standard for sending voice, video, and data over digital telephone lines or normal telephone wires. ISDN support data transfer rates of 64 Kbps (64,000 bits per second). Most ISDN lines offered by telephone companies give you two lines at once, called *B* channels. You can use one line for voice and the other for data, or you can use both lines for data to give you data rates of 128 Kbps

Intelligent Network (IN)

In an intelligent network (IN), the logic for controlling telecommunications services migrates from traditional switching points to computer-based, service-independent platforms. This provides network operators an open platform provisioned with generic service components that can interoperate with elements from different vendors, based on published, open-interface standards. This platform can be used to develop new and different services.

Intelligent Network Application Part (INAP)

A protocol layer that runs on top of TCAP. INAP is primarily a European standard, the equivalent to the AIN specification. The AIN and INAP specifications are similar and can be deployed using SS7 for functions such as call routing.

Internet Protocol (IP)

Connectionless datagram-oriented network layer protocol containing addressing et control information in the packet header that allows nodes to forward the packet toward the destination.

Internet telephony

Internet telephony refers to communications services – voice, facsimile, and/or voice-messaging applications – that are transported via the Internet, rather than the public switched telephone network (PSTN).

ISDN User Part (ISUP)

Defines the signalling functions needed to support services and user facilities for voice and non-voice applications in an ISDN environment.

ITU-T H.323

An ITU-T standard that defines how audio-visual conferencing data is transmitted across networks.

Java Telephony API (JTAPI)

Standard telephony application programming interface for computer telephone applications under Java. JTAPI provides the definition for a set of reusable telephone call control objects.

Lightweight Directory Access Protocol (LDAP)

Protocol used to communicate from a calling program (running on a node) and a directory node. Information is kept on the LDAP-based directory node about such topics as people and/or services.

Line signalling

Represents the intention to seize or release a local line (UNI) or a trunk circuit (NNI).

Local exchange

Local systems that provide local transmission services. It offers access to the user through the local loop and the digital trunk to tool offices.

Local loop

Pair of wires, or its equivalent, between a customer and its local exchange.

Location server

Device used to learn about the gateways in its domain as well as in other domains in order to provide this information to clients.

Message Transfer Part (MTP)

Lower layer of the SS7 architecture. Its objectives are to provide a reliable transfer of signalling messages between two "user" parts.

Multipoint Control Unit (MCU)

H.323 entity that provides the capability for three or more terminals participating in a multipoint conference. It may also connect two terminals in a point-to-point conference which may latter develop into a multipoint conference.

Out-band signalling

System where the call signalling is done independently of the data carrying. This allows the use of the full 64kbps bandwidth.

PC-to-PC call

Call established in a hybrid network by an Internet phone and intended to an Internet phone.

PC-to-phone call

Call established in a hybrid network by an Internet phone and intended to a traditional phone.

Phone-to-PC call

Call established in a hybrid network by a traditional phone and intended to an Internet phone.

Phone-to-phone call

Call established in a hybrid network by a traditional phone and intended to a traditional phone.

PINT (Service Call) Party

A person who is involved in a telephone network call that results from the execution of a PINT service request, or a telephone network-based resource that is involved (such as an automatic Fax Sender or a Text-to-Speech Unit).

PINT Client

An Internet host that sends requests for invocation of PINT Service, in accordance with [RFC2848].

PINT Executive System

A system that interfaces to a PINT Server and to a telephone network that executes a PINT service. It need not be directly associated with the Internet, and is represented by the PINT Server in transactions with Internet entities.

PINT Gateway

An Internet host that accepts requests for PINT Service and dispatches them onwards towards a telephone network.

PINT Requesting User

The initiator of a request for service. This role may be distinct from that of the "party" to any telephone network call that results from the request.

PINT Requestor

An Internet host from which a request for service originates.

PINT Service

A service invoked within a phone system in response to a request received from a PINT client.

Plain Old Telephone Services (POTS)

See Public Switch Telephone Network (PSTN).

Platform

Underlying hardware or software for a system. The platform defines a standard around which a system can be developed. Once the platform has been defined, software developers can produce appropriate software and managers can purchase appropriate hardware and applications.

Post Office Protocol (POP)

Protocol used to retrieve e-mail from a mail server. Most e-mail applications (sometimes called an e-mail client) use the POP protocol, although some can use the newer IMAP (Internet Message Access Protocol).

Proxy

An intermediary program that acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, possibly after translation, to other servers. A proxy interprets, and, if necessary, rewrites a request message before forwarding it.

PSTN/Internet INTernetworking (PINT)

Protocol which addresses connection arrangements through which Internet applications can request and enrich PSTN telephony services.

Public Switch Telephone Network (PSTN)

International telephone system based on copper wires carrying analog voice data. Today, it is almost entirely digital in technology except for the final link from the local telephone office to the user.

Pulse Code Modulation (PCM)

Technique for converting an analog signal with an infinite number of possible values into discrete binary digital words that have a finite number of values.

Quality of Service (QoS)

Ability to allocate network resources (bandwidth, buffer allocation, etc.) for communication between two stations.

Real-time Transport Protocol (RTP)

Protocol of the TCP/IP suite that conveys sequence numbers and timestamps between packet video and audio applications.

Register signalling

Represents the flow of dialled digits (i.e. the call destination) between two exchanges.

Resource reSeRvation Protocol (RSVP)

Protocol developed by the IETF to support different classes of service for IP flows.

Round robin scheme

Arrangement of choosing all elements in a group equally in some rational order, usually from the top to the bottom of a list and then starting again at the top of the list and so on. A simple way to think of round robin is that it is about "taking turns."

Service Access Point (SAP)

A point at which a designated service may be obtained.

Service Control Function (SCF)

A function that commands call control functions in the processing of IN provided and/or custom service requests. The SCF may interact with other functional entities to access additional logic or to obtain information (service or user data) required to process a call/service logic instance

Service Control Point (SCP)

An entity in the intelligent network that implements a service control function (SCF).

Service Creation Environment Function (SCEF)

The set of functions that support the service creation process. It allows to define, develop, test and input to SMF services to be provided in an intelligent network. Output of this function would include service logic, service management logic, service data template and service trigger information.

Service Creation Environment Point (SCEP)

A physical entity that implements a service creation environment function (SCEF).

Service Data Function (SDF)

Particular service aimed to be used as data server (and even object server) by other services. The SDF contains customer and network data for real time access by the SCF in the execution of an IN provided service. It does not necessarily encompass data provided by a third party, such as credit information, but may provide access to these data.

Service Data Point (SDP)

A physical entity that implements a service data function (SDF).

Service Execution Process (SEP)

Operational part of an IN service.

Service in the PSTN/IN Requesting InTernet Services (SPIRIT)

Protocol which addresses how services supported by IP network entities can be started from IN requests.

Service Independent Building Blocks (SIB)

Software entities that are service independent. These building blocks allow the developers of a wide variety of services to take advantage of the flexibility and reusability.

Service Level Agreement (SLA)

Service contract between a customer and a service provider that specifies the forwarding service a customer should receive.

Service Logic Execution Environment (SLEE)

The host that contains the SEP and the database for a given service. The SEPs for platform services are located in the SMFs, the SEPs for IN services in the SLEEs.

Service Management Function (SMF)

Functional entity involved with activities for service deployment, provisioning, control, monitoring and billing.

Service Management Point (SMP)

A physical entity that implements a service management function (SMF).

Service Switching Function (SSF)

Associated with the CCF, the SSF provides the set of functions required for interaction between the CCF and the SCF. Its major function is to invoke call and non-call related services on the IN platform.

Service Switching Point (SSP)

A physical entity that implements a service switching function (SSF).

Session Description Protocol (SDP)

Protocol intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation.

Session Initiation Protocol (SIP)

Application-layer control (signalling) protocol that provides the creation, modification and termination of sessions with one or more participants.

Signalling Connection Control Part (SCCP)

Group of additional transport related functions used to complement the functions of MTP.

Signalling End Point (SEP)

Local node in the network; source or sink of SS7 signalling traffic.

Signalling mode

Association between the path taken by a signalling message and the signalling relation to which the message refers. Examples of modes are associated mode, non-associated mode and quasi-associated mode.

Signalling Point

Node in an SS7 network. Examples of SPs are SEP or STP.

Signalling Transfer Point (STP)

SP that allows messages to travel from one SEP to another inside an SS7 network. Three levels of STP exist: National, International and Gateway.

SIP Initiator

The party initiating a conference invitation. Note that the calling party does not have to be the same as the one creating the conference.

SIP Transaction

A SIP transaction occurs between a client and a server and comprises all messages from the first request sent from the client to the server up to a final response sent from the server to the client. A transaction is identified by the CSeq sequence number within a single call leg.

SIP User Agent Client (UAC)

Client application that initiates the SIP request.

SIP User Agent Server (UAS)

Server application that contacts the user when a SIP request is received and that returns a response on behalf of the user. The response accepts, rejects or redirects the request.

Softswitch (software switch)

Generic term for any open application program interface (API) software used to bridge a public switched telephone network (PSTN) and voice over Internet protocol (VoIP) by separating the call control functions of a phone call from the media gateway (transport layer).

Specialised Resource Function (SRF)

Provides a category of resources for access by users. Examples can include DTMF sending and receiving, protocol conversion, speech recognition, synthesised speech provision, etc.

Specialised Resource Point (SRP)

A physical entity that implements a specialised resource function (SRF).

Stored Program Control (SPC)

A system whereby the instructions are placed in the memory of a common controlled switching unit and to which it refers while processing a call for instructions regarding class marks, code conversions, routing, as well as for trouble analysis.

Subscription Assignment Group (SAG)

Group of Parlay clients which have the same set of privileges.

Time Division Multiplexing (TDM)

A multiplexing technique whereby two or more channels are derived from a transmission medium by dividing access to the medium into sequential intervals. Each channel has access to the entire bandwidth of the medium during its interval. This implies that one transmitter uses one channel to send several bit streams of information.

Tool office

A center for the switching of toll calls.

Transaction Capabilities Application Part (TCAP)

Additional functions used for the support of remote operations in a real time environment.

Transmission Control Protocol (TCP)

Layer 4 transport protocol that reliably delivers packets to higher layer protocols. It performs sequencing and reliable delivery via retransmission.

Unified Modelling Language (UML)

General-purpose notational language for specifying and visualizing complex software, especially large, object-oriented projects.

User Datagram Protocol (UDP)

Layer 4 transport protocol that offers a limited amount of service when messages are exchanged between computers in a network that uses the Internet Protocol (IP).

Virtual Private Network (VPN)

A virtual private network utilizes a public network such as the Internet as a secure channel for communicating private data. VPN technology allows the creation of a secure link between a corporate LAN (local area network) and a remote user's PC.

Appendix 2

Acronyms

ACF	Admission Confirmation
ACM	Address Complete Message
ADSL	Asynchronous Digital Subscriber Line
AEG	Application Expert Group
AIN	Advanced Intelligent Network
ANM	Answer Message
API	Application Programming Interface
ARF	Admission Reject
ARQ	Admission Request
ATM	Asynchronous Transfer Mode
AVP	Audio/Video Profile
BCP	Basic Call Process
BCSM	Basic Call State Model
BGP	Border Gateway Protocol
CAS	Channel Associated Signalling System
CCAF	Call Control Agent Function
CCF	Call Control Function
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
CPE	Customer Premise Equipment
CS	Capability Set
CUIS	Call User Interaction Service
DNS	Domain Name System
DP	Detection Points
DPC	Destination Point Code
DPE	Distributed Processing Environment
DRF	Disengage Confirmation
DRQ	Disengage Request
DSCP	Differentiated Services Code Point

DSP	Destination Signalling Point
DSS1	Digital Subscriber Signalling System #1
DTMF	Dual Tone Multi Frequency
EF	Elementary Function
FDM	Frequency Division Multiplexing
FEA	Functional Entity Action
FEC	Forwarding Equivalence Class
GCCS	Generic Call Control Service
GSL	Global Service Logic
GTT	Global Title Translation
GTT	Global Title Translation
GUIS	Generic User Interface Service
HTTP	HyperText Transfer Protocol
IAM	Initial Address Message
IANA	Internet Assigned Number Authority
ICW	Internet Call Waiting
IETF	Internet Engineering Task Force
IF	Information Flow
IN	Intelligent Network
INAP	Intelligent Network Application Part
INCM	Intelligent Network Conceptual Model
IP	Internet Protocol
IPX	Internet Packet Exchange
ISDN	Integrated Service Digital Network
ISP	Internet Service Provider
ISUP	ISDN User Part
ITU	International Telecommunication Union
ITU-T	Telecommunication standardization sector of ITU
JAIN	Java APIs for Integrated Networks
JCAT	Java Coordination And Transaction
JCC	Java Call Control
JSLEE	Java Service Logic Execution Environment
JTAPI	Java Telephony API
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol

MAN	Metropolitan Area Network
MCU	Multipoint Control Unit
MFC	Multi-frequency
MG	Media Gateway
MGC	Media Gateway Control
MGCU	Media Gateway Control Unit
MGU	Media Gateway Unit
MIME	Multipurpose Internet Mail Extensions
MPLS	Multi Protocol Label Switching
MTP	Message Transfer Part
NAT	Network Address Translator
NGN	Next Generation Network
NNI	Network-to-Network Interface
NSP	Network Service Part
NSP	Network Service Part
OA&M	Operation Administration and Management
ODP	Open Distributed Processing
OO	Object Oriented
OPC	Originating Point Code
OSI	Open System Interconnection
OSP	Originating Signalling Point
OSP	Open Service Platform
PBX	Private Branch Exchange
PCM	Pulse Code Modulation
PE	Physical Entity
PEG	Protocol Expert Group
PIC	Points In Call
PINT	PSTN/Internet INTernetworking
POI	Points Of Initiation
POP	Post Office Protocol
POR	Points Of Return
POTS	Plain Old Telephone Services
PSTN	Public Switch Telephone Network
QoS	Quality of Service
R2C	Request To Call

R2F	Request To Fax
R2HC	Request to Hear Content
RAS	Registration, Admission and Status
RAS	Registration, Administration and Status
REL	Release
RLC	Release Complete
RSVP	Resource reSeRvation Protocol
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
SAC	Service Access Code
SAG	Subscription Assignment Group
SAP	Session Announcement Protocol
SAP	Service Access Point
SCCP	Signalling Connection Control Part
SCE	Service Creation Environment
SCF	Service Control Function
SCN	Switched Circuit Network
SCP	Service Control Point
SCTP	Stream Control Transmission Protocol
SDF	Service Data Function
SDP	Service Data Point
SDP	Session Description Protocol
SEP	Signalling End Point (SS7-related)
SEP	Service Execution Process (IN-related)
SF	Service Feature
SG	Signalling Gateway
SGU	Signalling Gateway Unit
SIB	Service Independent building Block
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SLEE	Service Logic Execution Environment
SMF	Service Management Function
SMP	Service Management Point
SMS	Short Message Service

SP	Signalling Point
SPA	Service Provider Access
SPC	Stored Program Control
SPIRIT	Service in the PSTN/IN Requesting InTernet Services
SRF	Specialised Resource Function
SROBJ	Send and Receive OBJect
SRP	Specialised Resource Point
SS7	(Common Channel) Signalling System number 7
SSF	Service Switching Function
SSP	Service Switching Point
STP	Signalling Transfer Point
TCAP	Transaction Capabilities Application Part
TDM	Time Division Multiplexing
TRIP	Telephony Routing Information Protocol
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
UML	Unified Modelling Language
UNI	User-to-Network Interface
URI	Universal Ressource Identifier
URL	Uniform Resource Locator
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
WAN	Wide Area Network
WAP	Wireless Application Protocol

Appendix 3

Bibliography

[ADC99] ADC Telecommunications. *NewNet SS7 Tutorial.* Available from http://www.iec.org/tutorials, 1999.

[ALBT] *Web Call Completion with « Browse & Talk ».* Alcatel White Paper. Date not given.

[ALCA98] *General Telecom, Common Channel Signaling #7.* Internal document, Alcatel, 1998.

[ALCA00a] *IN Concepts.* Internal document. Alcatel, 2000.

[ALCA00b] *General Introduction release 2.3.* Internal document. Alcatel, 2000.

[ALCA00c] *Telecom Concepts 2000.* Internal document. Alcatel, 2000.

[ALCA00d] *Intelligent Networks Glossary release 2.3.* Internal document. Alcatel, 2000.

[APION] http://www.apion-tss.com/CM/Services/intnet.html.

[BGD00] S. Beddus, G. Bruce, S. Davis. *Opening Up Networks with JAIN Parlay.* IEEE Communication Magazine, April 2000, pp. 136-143.

[BLAC00] U. Black. *Internet Telephony: Call Processing Protocols.* Prentice Hall Series in Advanced Communications Technologies, 2000.

[CGI] http://hoohoo.ncsa.uiuc.edu/cgi/interface.html.

[DARE00] B. Davie and Y. Rekther. *MPLS technology and applications.* Morgan Kaufmann, 2000.

[DPJ00] A. Deo, K. Porter, M. Johnson. *Java SIP Servlet API Specification.* April 2000, available from http://www-uk.hpl.hp.com/people/sth/sip/servlet/

[E164] International Telecommunication Union. *The International Public Telecommunication Numbering Plan*, Recommendation E.164. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1997.

[FGL00] I. Faynberg, L. Gabuzda and H. Lu. *Converged Networks and Services: Internetworking IP and the PSTN.* Wiley, 2000.

[G711] International Telecommunication Union. P*ulse code modulation (PCM) of voice frequencies*, Recommendation G.711. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, November 1988.

[G729] International Telecommunication Union. *C source code and test vectors for implementation verification of the G.729 8 kbit/s CS-ACELP speech coder*, Recommendation G.729. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, March 1996.

[GLI00] R. Glitho. Advanced Services Architectures for Internet Telephony: A critical Overview. IEEE Network, July/Augustus 2000.

[GRAP01] S. Grapta. *The Role Of The Softswitch In The Next-Generation Network*. TMCNET, 2001. Available from http://www.tmcnet.com/it/0301/0301acc.htm.

[H225.0] International Telecommunication Union. *Call Signalling Protocols and Media Stream Packetization for Packet-based Multimedia Telecommunication Systems*, Recommendation H.225.0. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, September 1999.

[H245] International Telecommunication Union. *Control Protocol for Multimedia Communication,* Recommendation H.245. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, September 1999.

[H323] International Telecommunication Union. *Packet-based Multimedia Communication Systems,* Recommendation H.323. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, September 1999.

[H450.1] International Telecommunication Union. *Generic functional protocol for the support of supplementary services in H.323*, Recommendation H.450.1. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, October 1998.

[HAN] H.E. Hanrahan. *Evolution of Standards for Advanced Telecommunications Services and Network Management.* Centre for Telecommunication and Services, University of the Witwatersand, Johannesburg. Date not given.

[IECCN] The International Engineering Consortium. *Convergence Switching and the Next-Generation Carrier.* Available from http://www.iec.org/tutorials, date not given.

[IECIPIN] The International Engineering Consortium. *Internet Protocol (IP) / Intelligent Network (IN) Integration.* Available from http://www.iec.org/tutorials, date not given.

[IECSOFT] The International Engineering Consortium. *Local-Exchange Softswitch System: Softswitch and Packet Voice.* Available from http://www.iec.org/tutorials, date not given.

[IECSS7] The International Engineering Consortium. *Signaling System 7 (SS7).* Available from http://www.iec/org/tutorials, date not given.

[IECSS7GW] The International Engineering Consortium. *Signaling System 7 (SS7) Gateway Solution for Internet Access.* Available from http://www.iec/org/tutorials, date not given.

[JAWP01] SUN Microsystems. *The JAIN APIs: Integrated Networks APIs for the Java Platform. A white paper describing the JAIN objectives, overall technical architecture and program structure.* January 2001, available from http:/java.sun.com/products/jain.

[JCC00] R. Jain. F. Anjum, P. Missier, S. Shastry. *Java Call Control, Coordination, and Transactions.* IEEE Communication Magazine, January 2000, pp. 108-114.

[JKDT99] J. de Keijzer, D. Tait. *Will The Intelligent Network Still Matter In An IP-Centric World?* CTI Network Telephony, April 1999. Available from http://www.tmcnet.com/articles/ctimag/0499/0499sun.htm.

[JTAPI] http://java.sun.com/products/jtapi.

[JRRS00] J. Rosenberg, R. Shockey. *A Key Component for Internet Telephony.* Computer Telephony, Volume 8, Issue 6, June 2000.

[KMS98] J. Kozik, W. Montgomery, J. Stanaway. *Voice Services in Next-Generation Networks: The Evolution of the Intelligent Network and Its Role in Generating New Revenue Opportunities.* Bell Labs Technical Journal, October-December 1998, pp 124-143.

[LAKS99] Ramnath A. Laksmi-Ratan. *The Lucent Technologies Softswitch – Realizing the Promise of Convergence.* Bell Labs Technical Journal, April-June 1999, pp 174-195.

[KTG00] J. de Keijzer, D. Tait, R. Goedman. *JAIN: A New Approach to Services in Communication Networks.* IEEE Communication Magazine, January 2000, pp. 94-99.

[LUC00a] J-L. Bakker et al. *Rapid Development and Delivery of Converged Services Using APIs.* Lucent Technologies, Bell Labs Technical Journal, July-September 2000, pp. 12-29.

[LUC00b] C. Bear, W. Montgomery, D. Nolte, S. Rajchel, M. Silva. *Open, Programmable Networks.* Lucent Technologies, Bell Labs Technical Journal, July-September 2000, pp. 30-42.

[LUC00c] M. Grech and al. *Delivering Seamless Services in Open Networks Using Intelligent Service Mediation.* Lucent Technologies, Bell Labs Technical Journal, July-September 2000, pp. 186-202.

[LUC00d] P. Sijben, J. Segers, L. Spergel, J. Kozik. *Building the Bridge: Devising an Architecture to Migrate Voice-Band Calls to Packet Transport and Multimedia Services.* Lucent Technologies, Bell Labs Technical Journal, July-September 2000, pp. 166-185.

[LUC00e] J. Kozik, W. Montgomery, J. Stanaway. *Voice Services in Next-Generation Networks: The Evolution of the Intelligent Network and Its Role in Generating New Revenue Opportunities.* Lucent Technologies, Bell Labs Technical Journal, July-September 2000, pp. 124-143.

[LUC00f] D. Dowden, K. Kocan, J. Kozik. *The Future of Network-Provided Communications Services.* Lucent Technologies, Bell Labs Technical Journal, July-September 2000, pp. 3-11.

[LUC98] H. Schulzrinne, J. Rosenberg. *The Session Initiation Protocol: Providing Advanced Telephony Services Across the Internet.* Lucent Technologies, Bell Labs Technical Journal, October-December 1998, pp. 144-160.

[MPLSWG] MPLS Working Group, available from http://www.ietf.org/html.charters/mpls-charter.html.

[NAT00] B. Biggs. *A SIP Application Level Gateway for Network Address Translation.* Internet Draft, Internet Engineering Task Force, March 2000. Work in progress.

[PARLAY] Parlay API specification, available from http://www.parlay.org/specifications.

[PARWP] The Parlay Consortium. *What is Parlay?* June 2000, available from http://www.parlay.org.

[PASP00a] The Parlay Consortium. *Parlay APIs 2.1: Framework Interfaces, Client Application View.* June 2000.

[PASP00b] The Parlay Consortium. *Parlay APIs 2.1: Framework Interfaces, Parlay Service View.* June 2000.

[PASP00c] The Parlay Consortium. *Parlay APIs 2.1: Framework Sequence Diagrams.* June 2000.

[PASP00d] The Parlay Consortium. *Parlay APIs 2.1: Call Processing Sequence Diagrams.* June 2000.

[PMIB00] M. Krishnaswamy, D. Romascanu. *Management Information Base for the PINT Services Architecture.* Internet Draft, Internet Engineering Task Force, November 2000. Work in progress.

[PTWG] PINT working group, available from http://www.ietf.org/html.charters/pint-charter.html.

[PW99] B. Pagurek, T. White, *A Quick evaluation of H.323/H.450.* SCE Technical Report, February 1999.

[Q700] International Telecommunication Union. *Introduction to CCITT Signalling System No. 7*, Recommendation Q.700. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, March 1993.

[Q702] International Telecommunication Union. *Signalling Data Link*, Recommendation Q.702. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, 1993.

[Q703] International Telecommunication Union. *Signalling Link*, Recommendation Q.703. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, July 1996.

[Q704] International Telecommunication Union. *Signalling Network Functions and Messages*, Recommendation Q.704. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, July 1996.

[Q931] International Telecommunication Union. *ISDN user-network interface layer 3 specification for basic call control*, Recommendation Q.931. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1998.

[Q1200] International Telecommunication Union. *General series Intelligent Network Recommendation structure*, Recommendation Q.1200. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, September 1997.

[Q1201] International Telecommunication Union. *Principles of Intelligent Network architecture*, Recommendation Q.1201. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, October 1992.

[Q1202] International Telecommunication Union. *Intelligent Network – Service plane architecture*, Recommendation Q.1202. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, September 1997.

[Q1203] International Telecommunication Union. *Intelligent Network – Global functional plane architecture*, Recommendation Q.1203. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, September 1997.

[Q1204] International Telecommunication Union. *Intelligent network distributed functional plane architecture*, Recommendation Q.1204. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1993.

[Q1205] International Telecommunication Union. *Intelligent network – Physical plane architecture*, Recommendation Q.1205. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, March 1993.

[Q1218] International Telecommunication Union. *Interface recommendation for Intelligent Network CS-1*, Recommendation Q.1218. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, October 1995.

[RBRG00] R. Bhat, R. Gupta. *JAIN Protocol APIs.* IEEE Communication Magazine, January 2000, pp. 100-107.

[RFC1771] Y. Rekhter and T. Li. *A Border Gateway Protocol 4 (BGP-4).* Internet RFC 1771, March 1995.

[RFC1889] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications. Audio-Video.* Internet RFC 1889, January 1996.

[RFC1890] H. Schulzrinne. *RTP Profile for Audio and Video Conferences with Minimal Control.* Internet RFC 1890, January 1996.

[RFC2068] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. *Hypertext Transfer Protocol -- HTTP/1.1.* Internet RFC 2068, January 1997.

[RFC2231] N. Freed, K. Moore. *MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations.* Internet RFC 2231, November 1997.

[RFC2326] H. Schulzrinne, A. Rao, R. Lanphier. *Real Time Streaming Protocol (RTSP).* Internet RFC 2326, April 1998.

[RFC2327] M. Handley, V. Jacobson. *SDP: Session Description Protocol.* Internet RFC 2327, April 1998.

[RFC2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss. *An Architecture for Differentiated Service.* Internet RFC 2475, December 1998.

[RFC2543] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg. *SIP: Session Initiation Protocol.* Internet RFC 2543, March 1999.

[RFC2663] P. Srisuresh, M. Holdrege. *IP Network Address Translator (NAT) Terminology and Considerations.* Internet RFC 2663, August 1999.

[RFC2719] L. Ong et al. *Framework Architecture for Signaling Transport.* Internet RFC 2719, October 1999.

[RFC2960] R. Stewart and al. *Stream Control Transmission Protocol.* Internet RFC 2960, October 2000.

[RFC2976] S. Donovan. *The SIP INFO Method.* Internet RFC 2976, October 2000.

[RFC2995] H. Lu and al. *Pre-Spirits Implementations of PSTN-initiated Services.* Internet RFC 2995, November 2000.

[RFC3050] J. Lennox, H. Schulzrinne and J. Rosenberg. *Common Gateway Interface for SIP.* Internet RFC 3050, January 2001.

[RLS99] J. Rosenberg, J. Lennox, H. Schulzrinne. *Programming Internet Telephony Services.* Columbia University, New York, Technical Report CUCS-010-99, March 1999. Available from ftp://ftp.cs.columbia.edu/reports/reports-1999/cucs-010-99.ps.gz.

[RSGL01] J. Rosenberg, H. Schulzrinne. *Guidelines for Authors of SIP Extensions.* Internet Draft, Internet Engineering Task Force, March 2001. Work in progress.

[SAP96] M. Handley. *SAP: Session announcement protocol.* Internet Draft, Internet Engineering Task Force, November 1996. Work in progress.

[SFLW01] L. Slutman, I. Faynberg, H. Lu, M. Weissman. *The SPIRITS Architecture.* Internet Draft, Internet Engineering Task Force, April 2001. Work in progress.

[SIGWG] SIGTRAN Working group, available from http://www.ietf.org/html.charters/sigtran-charter.html.

[SIJB00] P. Sijben et al. *Building the Bridge: Devising an Architecture to Migrate Voice-Band Calls to Packet Transport and Multimedia Services.* Bell Labs Technical Journal, July-September 2000, pp 166-185.

[SIPEX01] A. Johnston, R. Sparks, C. Cunningham, S. Donovan and K. Summers. *SIP Service Examples.* Internet Draft, Internet Engineering Task Force, March 2001. Work in progress.

[SIP-T00] A. Vemuri, J. Peterson. *SIP for Telephones (SIP-T): Context and Architectures.* Internet Draft, Internet Engineering Task Force, November 2000. Work in progress.

[SPWG] SPIRITS working group, available from http://www.ietf.org/html.charters/spirits-charter.html.

[SR98] H. Schulzrinne, J. Rosenberg. *A comparison of SIP and H.323 for Internet telephony.* Network and Operating System Support for Digital Audio and Video (NOSSDAV), Cambridge, England, July 1998.

[SSCT] The Soft switch consortium, see http://www.softswitch.org.

[SUN00] http://www.sun.com/smi/Press/sunflash/2000-06/sunflash.20000606.28.html.

[TANE96] Andrew S. Tanenbaum. *Computer Networks: Third Edition.* Prentice Hall PTR, New Jersey, 1996.

[TELECO] Telecordia Technologies. *Intelligent Network (IN) tutorial.* Available from http://www.iec.org/tutorials/in, date not given.

[TRIP00] J. Rosenberg, H. Salama, M. Squire. *Telephony Routing over IP (TRIP).* Internet Draft, Internet Engineering Task Force, July 2000. Work in progress.

[VOIP99] U.Black. *Voice Over IP.* Prentice Hall Series in Advanced Communications Technologies, 1999.

[X213] International Telecommunication Union. *Information Technology – Open Systems Interconnection – Network Service Definition*, Recommendation X.213. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, November 1995.

[X950] International Telecommunication Union. *Information Technology – Open Distributed Processing – Trading Function: Specification*, Recommendation X.950. Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Augustus 1997.

Appendix 4

Common Channel Signalling System #7

In the past years, telecommunication networks have changed enormously. The single telephone network is evolving into a more generic network, capable of transporting all kinds of information. In order to manage this changing environment, a flexible signalling system is required, which can perform signalling functions for all telecom applications, existing or not yet defined. Common Channel Signalling System number 7 (SS7 for short) is the answer to these requirements. This appendix is based on an excellent document from Alcatel (see [ALCA98]). We invite you to consider this appendix as a short description of SS7 and not as an extensive and critical view of this protocol.

A4.1 Topology of signalling systems

A telecommunication network is divided in two parts: the access network composed of local loops and CPE's, and the core network (also called the backbone) composed of the switching offices. The functions needed to transport information on a telecommunication network can be subdivided into two groups:

- The transmission itself over a physical line from an origin to a destination
- Before transmitting any information, a leg must be attached from the origin to the destination. This operation is called the switching

We have shortly seen; in chapter 2, concepts relating to the transmission. This appendix is devoted to the switching functionality in telephone networks. In order to perform the switching function, two types of communication are required. First, a communication between the calling party and his own switching unit: this is the User-to-Network interface (UNI). Secondly, a communication between each switching unit and the next one in the call sequence: this is the Network-to-Network interface (NNI). These concepts are schematised in Figure A4.1.

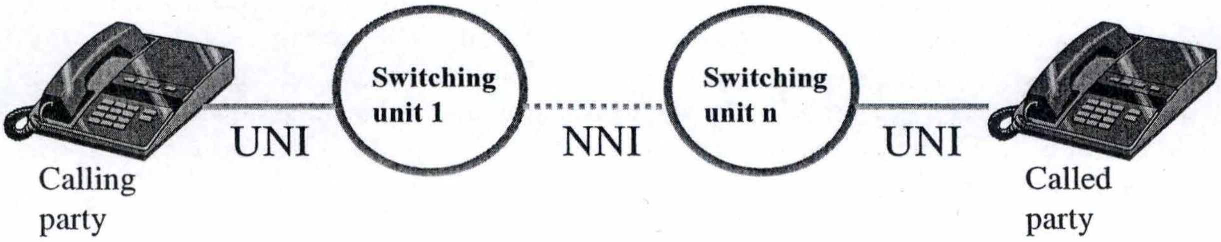


Figure A4.1 – UNI and NNI

Well-known examples of UNI are Digital Subscriber Signalling System #1 (DSS1) also called "D-channel protocol" (see [Q931] for the layer 3 functionality) or simply analog tones for the majority of users. What concerns NNI, two main signalling systems exist: Channel Associated

Signalling System (CAS) and Common Channel Signalling System #7 (SS7) who is the most widely used.

Another method can be used to classify signalling system, i.e. classifying according the functionality. At least two types of information must be exchanged between two points in the network:

- Line signalling represents the intention to seize or release a local line (UNI) or a trunk circuit (NNI)
- Register signalling represents the flow of dialled digits (i.e. the call destination) between two exchanges

A4.2 History of signalling systems

This history of signalling systems is intended to introduce various concepts associated with signalling and switching. Two main "class" of signalling systems have been used before SS7: analogue signalling and Channel Associated Signalling system (CAS).

Analogue signalling is an in-band signalling, that is to say that wire that is used for conversation is also used for signalling. Such signalling system consists of analog electrical representations of sound. In fact, these signals are sent over the wire in the form on an analog current that are converted back to sound at the receiving end. The same way is used to transport voice traffic. In such system, the register signalling uses Multi-frequency (MFC) tones to represent digits. Analogue signalling has a lot of limitations. The very first one is the limited number of states that can be represented by electric signals. Thus, it fails to provide advanced telephonic-features like the Intelligent Network features of today.

Channel Associated Signalling (CAS) is a digital signalling system who generally uses 2Mbit/sec PCM-link. The CAS frame is structured into 32 channels of 64kbps each. CAS is also an in-band signalling: channel 16 is used for signalling and channel 0 is used for synchronization. The line signalling is transported in a structure called Multi-frame. The first frame of the channel 16 contains multi-frame synchronization information. Then, each following frames contain information related to 2 channels (4 bits for each channel). Thus in the second frame, there is information related to channel 1 and 17 while the third frame contains information about channel 2 and 18, etc. 4-bits information can be IDLE (1001) and ACK (1101) to cite a few. Note that CAS performs line signalling only. Register signalling is carried in the user channel itself in a digitalized way. The multi-frame concept is the main limitation of the CAS line signalling system. Indeed, the period of a multi-frame is 2 msec (16 * 125 µsec). Thus in two directions, the line signalling alone can take up to 4msec[1]. CAS line signalling, completed with register signalling over the user channels, is a too inefficient signalling system for the requirements of today telecommunication networks.

Common Channel Signalling System #7 was developed in order to satisfy the need for a common signalisation interface in national and international borders. This signalling system was standardized by the ITU-T (Recommendations series Q.7xx). We described it shortly in the remainder of this appendix. SS7 is an NNI signalling system that performs both Line and Register signalling.

[1] The reason for this is that the sender does not choose in which frame he transmits his information. In a worst case, the sender has to wait event if there is no other signaling information to be sent.

A4.3 SS7: definition, objectives and properties

A4.3.1 Definition

A good definition of SS7 can be found in [ADC99]: "Signalling System No. 7 (SS7) is a global standard for telecommunications defined by the ITU-T. It defines the procedures and protocol by which network elements in the Public Switch Telephone Network (PSTN) exchange information over a digital signalling network to effect wireless and wireline call setup, routing and control".

A4.3.2 Objectives

Belonging to [Q700], the objective of SS7 is to provide an internationally standardized, general purpose, common channel signalling system that is optimised for operation in digital telecommunications network, at least over 64 kbps digital channels. It must also provide a reliable means of information transfer in support of call control; remote control; and operations, management and administration. Moreover, SS7 must provide a reliable means to transfer signalling information in correct sequence and without loss or duplication.

A4.3.3 Properties

First, SS7 offers out-band signalling, that is to say that control information and user information are logically separated. However, it does not mean that this information is separated physically. Out-band signalling establishes a separate digital channel for the exchange of signalling information. Such channel is called a signalling link. The advantages of using out-band signalling in place of in-band signalling are:

- It allows the transport of more data at higher speed
- The possibility to extend the signalling to multimedia
- It allows for signalling at any time of the call
- Voice links remain idle, until the called party answers the call

Secondly, SS7 is message-oriented. This feature allows converting line signalling or register signalling into an information message. One could imagine the possibilities of such approach. Effectively, we are far away of the analogue signalling system limited by the number of states that can be represented by electric signals.

Finally, the signalling channel used between two exchanges is a common channel, that is to say one signalling channel can carry about 1000 voice channels. This improvement in signalling system required, however, extra measures like ordered delivery of the signalling packets and addressing information, i.e. which voice channels the signalling is about. The order delivery of the signalling packets is achieved by the signalling modes of SS7. A signalling mode is an association between the path taken by a signalling message and the signalling relation to which the message refers. Three signalling modes exist: in the first, associated mode, the messages relating to a particular signalling relation between two adjacent points are conveyed over a link set, directly interconnecting these signalling points[2]. In the second, non-associated mode, the messages are conveyed over two or more link sets passing through one or more signalling points other than those which are the origin and the destination of the messages. The last one, quasi-associated mode, is a restriction of the non-associated mode where the path taken by the message through the signalling network are pre-determined and, at a given point in time, fixed. SS7 works in associated and quasi-associated mode. In fact, lowered layers of SS7 do not provide features to avoid out-of-sequence arrival of messages but the signalling mode avoid these problems that would typically arise in a fully non-associated mode with dynamic message routing.

A4.4 SS7 network architecture

In this section, a typical SS7 structure is discussed. SS7 is logically separated from any other networks. All nodes in the SS7 network are called Signalling Points (SP). Examples of Signalling Points are switch exchange or Service Control Point (SCP) used in Intelligent Networks (see chapter 2).

In the field of SS7, two kinds of Signalling Points must come to the fore. First, Signalling End Point (SEP) is a local node in the network; it is a source or a sink of signalling traffic. A typical example of a SEP is a Service Switching Point described in chapter 2. Secondly, Signalling Transfer Point (STP) allows messages to travel from one SEP to another. In an STP, there is no switching-oriented processing, that is to say the content of the signalling messages is not examined. Three levels of STP exist: National, International and Gateway. SEPs and STPs are schematised in Figure A4.2.

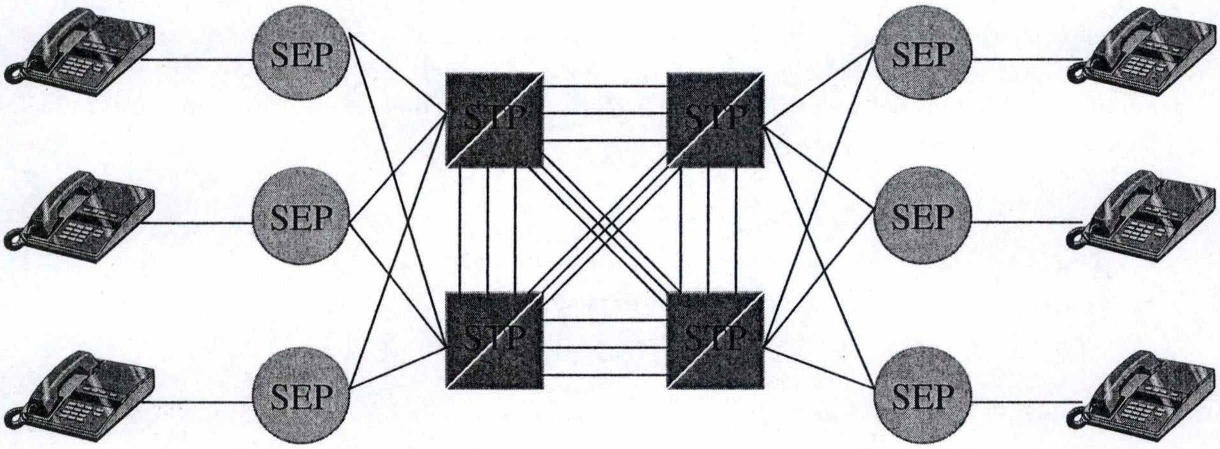


Figure A4.2 – SS7 architecture

STPs are often deployed in pairs for redundancy. To improve the reliability of the network, alternate and multiple paths are deployed between SPs: SEPs are connected to at least two STPs while multiple paths exist between different STPs. Before studying the SS7 protocol, we give three more definitions. Originating Signalling Point is an SP at which a message is generated while a Destination Signalling Point is an SP at which a message is destined. Each SP is identified with a point code. The point code identifies an SP in a unique way inside the SS7 network.

[2] Signalling Points are defined in the next section.

A4.5 SS7 protocol overview

In this section, we give a short overview of the SS7 protocol. To achieve the needed requirements, SS7 must be flexible and modular. Like many other protocols, it has a layered structure that can be compared to the OSI[3] model. The fundamental principle of SS7 is the division of functions into a Message Transfer Part (MTP) on one hand, and separate users parts on the other hand. Please note that the term "user" in this context refers to any entity that utilizes the MTP functionalities.

A4.5.1 SS7 structure

The Message Transfer Part (MTP) objectives are to provide a reliable transfer of signalling messages between two "user" parts. Moreover, MTP also provides flow control functionality. On top of MTP, several user parts can run: each user part supports a specific application. The message exchanged between two "user" parts are standardised. With the evolution of the network utilizations, MTP become insufficient to cope with new requirements (for instance, the possibility to address application within an SP). Signalling Connection Control Part (SCCP) is a new layer that increases the functionalities of the transport system. Transaction Capabilities Application Part (TCAP), which is used by INAP, is the user part of SCCP that defines the messages and protocol used to communicate between applications in nodes. All these concepts are schematised in Figure A4.3.

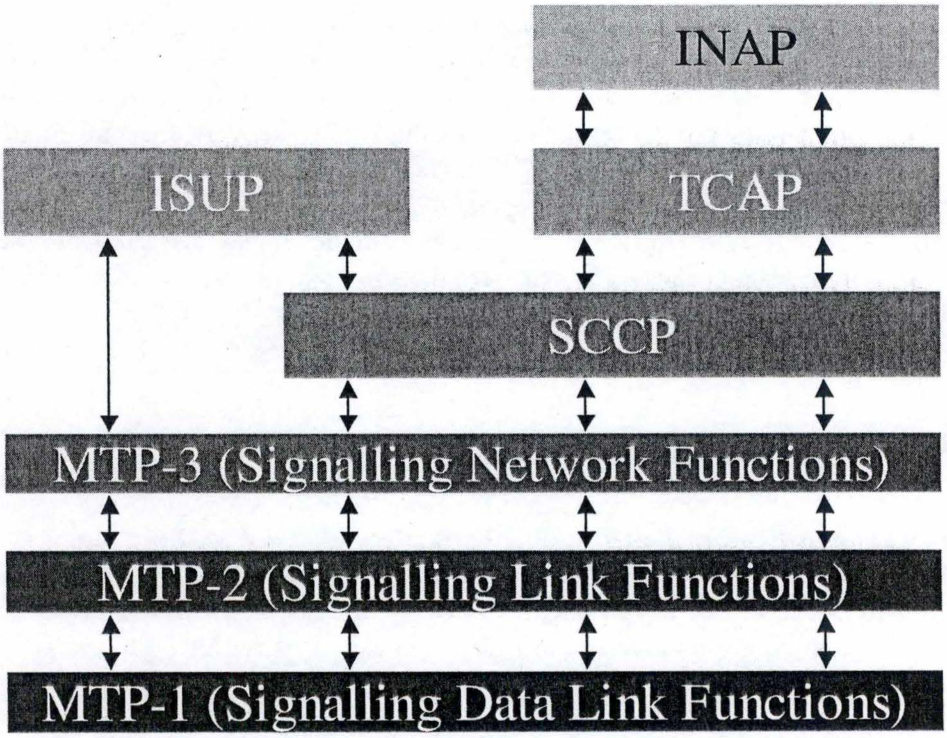

INAP
ISUP
TCAP
SCCP
MTP-3 (Signalling Network Functions)
MTP-2 (Signalling Link Functions)
MTP-1 (Signalling Data Link Functions)

Figure A4.3 – SS7 structure

[3] Open System Interconnection.

A4.5.2 Message Transfer Part – Level 1

The Signalling Data Link Functions provide a bearer for a signalling link. SS7 makes use of existing bit-transport equipment. The channel used to carry SS7 information (that is signalling link) must be used exclusively by SS7 entities. No other information can be carried together with the signalling information in the same channel. Note again that SS7 is optimised to operate in a digital environment but it can be used in an analogue environment at lower speeds as well. A good description of MTP-1 can be found in [ALCA98, pp. 38-41].

A4.5.3 Message Transfer Part – Level 2

While MTP-1 transmits messages from exchange to exchange, MTP-2 provides reliable transfer of signalling messages between two directly connected signalling points. MTP-2 checks that messages are free of errors and that no information is lost during transmission. Signalling messages provided by higher levels are transmitted over the signalling link in variable length Signal Units. These Signal Units contain, in addition to signal information, transfer control information. This transfer control information allows providing error detection and correction; signalling link error monitoring and flow control. A complete description of MTP-2 can be found in [ALCA98, pp. 42-67].

A4.5.4 Message Transfer Part – Level 3

MTP-3 consists of two parts: the first, signalling message handling functions, ensures that signalling messages are delivered to the proper user part. The Destination Point Code (DPC) represents the point code of the destination exchange while the Originating Point Code (OPC) represents the point code of the originating exchange. This delivery may be made directly or indirectly, via one or more intermediate STPs. The second, signalling network management functions, provides reconfiguration of the signalling network in the case of failures and control traffic in case of congestion. Furthermore, theses functions can also activate or align new signalling links. Then, when failures are restored, the opposite actions can take place in order to re-establish normal signalling activity. A complete description of MTP-3 can be found in [ALCA98, pp. 68-101].

A4.5.5 Signalling Connection Control Part

Signalling Connection Control Part (SCCP) is the answer to some shortcomings of the MTP stack: the first is the impossibility to reach destinations in another country[4]. The second is that MTP does not provide connection-oriented services. The last one is that MTP is not fully OSI-compatible[5]. The combination of MTP and SCCP is called Network Service Part (NSP). NSP provides a subset of the layer 3 services defined in [X213]. Especially, SCCP provides, on one hand, a more flexible routing scheme. As a result, routing of messages between any exchanges is supported, even if they are located in different countries. This feature is achieved by the Global

[4] This problem is due to the DPCs contained in routing tables. DPCs are coded on 14 bits. It means that the range of the DPCs is 2^{14} = 16384. Obviously, this is not enough to provide a unique worldwide identification of all exchanges using SS7.

[5] The connection-oriented version of SCCP is described in the form of OSI primitives and thus can be accessed by any OSI application.

Title Translation (GTT). On the other hand, SCCP provides a connection-oriented protocol who makes routing of multiple data messages to the same destination much more efficient. A complete description of SCCP can be found in [ALCA98, pp. 102-124].

A4.5.6 User Parts

Considering SCCP and MTP as a whole, SS7 consists of two basic modules: first a data transport system, which uses the digital telephone network to transport signalling information. Then, a number of different user parts. Each of them contains a set of messages and procedures describing the use of these messages. For example, ISDN User Part (ISUP) defines the signalling functions needed to support services and user facilities for voice and non-voice applications in an ISDN environment. "The ISUP is also suited for application in dedicated telephone and circuit-switched data networks and in analogue, and mixed analogue/digital networks" [Q700, p. 13]. Transaction Capabilities Application Part (TCAP) provides the means to establish non-circuit-related communication (unlike ISUP) between two nodes in the signalling network. Finally, Intelligent Network Application Part (INAP) runs on top of TCAP and has been overviewed in chapter 2. We illustrate, in Figure A4.4, the utilization of ISUP as an NNI for a call establishment between two analogue telephones.

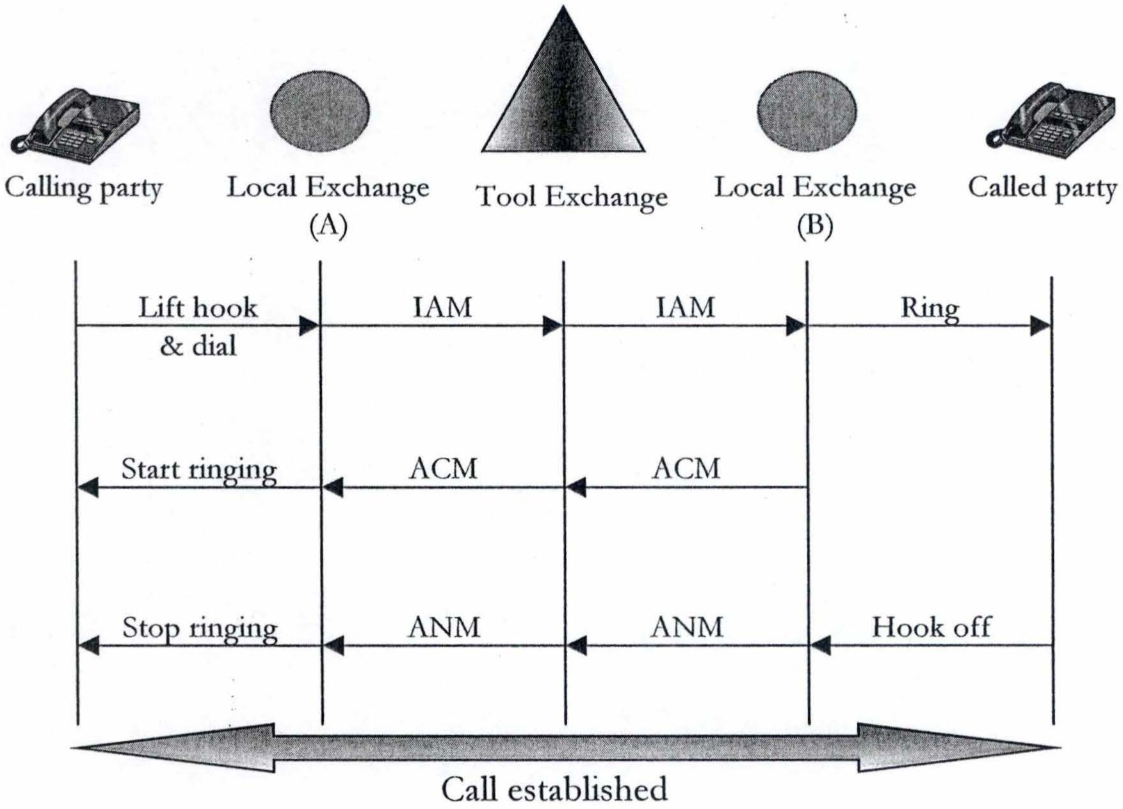


Figure A4.4 – Call establishment using ISUP

This figure represents the concepts that have been previously schematised in Figure 1.1. First of all, the calling party dials the number of the called party. A analyses the dialled digits and determines to which exchange it needs to send the call (here B). To do so, it selects an idle trunk between itself and B and formulates an Initial Address Message (IAM)[6]. The IAM message is used to initiate seizure of an outgoing circuit and to transmit calling party and called party numbers as well as other information relating to the routing and handling of the call. Once B receives the IAM message, it checks it serves the called party and that the called party is idle. Then B formulates an Address Complete Message (ACM) to indicate that the IAM message has reached its right destination. Simultaneously, the called party's CPE begins to ring. On receiving the ACM, A connects the calling local loop to the previously selected trunk, so that the caller can hear the ringing tones sent by B. When the called party picks up the phone, B formulates an Answer Message (ANM) to indicate that the call has been answered. The charging and the measurement of the call duration can start. The called and the calling parties are connected. The release of the call is performed in the same way with other ISUP messages, i.e. Release (REL) and Release Complete (RLC).

A4.6 SS7: conclusion

SS7 is a signalling system designed for use in a computer controlled telecommunication network. Two main properties have been underlined: flexibility and modularity. These properties involve that the system is flexible enough to adapt to evolutions in telecommunication networks. To provide IN services on the Internet, it was critical to have such system available. In chapter 3, we discuss how interaction can be achieved between IP and the PSTN for one thing by using SS7.

[6] The way SS7 messages are routed through the network (here by a tool exchange) is avoided by simplicity.

Appendix 5

Signalling standards for Internet telephony

In order to provide useful services, Internet telephony requires a set of control protocols for connection establishment, capabilities exchange, and conference control. In this appendix we overview and compare two standards that have emerged for signalling and control for Internet telephony. The first is the ITU-T Recommendation H.323 (see [H323]) and the other is Session Initiation Protocol (SIP see [RFC2543]) standardized by the IETF. SIP and H.323 resolve the same problem in a different way. Indeed, H.323 uses the traditional circuit-switched approach to signalling (mainly the ITU-T recommendation Q.931 related to the ISDN environment) while SIP privileges more the lightweight Internet approach based on HTTP. From this analysis, we shall see that a huge number of protocols have emerged in this domain: this is the reason why we try to find out which of these two protocols is best suited to meeting our requirements, i.e. that can provide IN features on an IP network.

A5.1 H.323

H.323 is a standard that specifies the components (see section A5.1.1), protocols (see section A5.1.2) and procedures (see section A5.1.3) that provide real-time audio, video and data communications over any packet-based network, for example the IP network. H.323 can also be applied to multicast-multipoint applications. The first version of H.323 was designed to provide multimedia conferencing in a LAN environment with no QoS associated. The emergence of VoIP applications led to a standardisation for IP telephony. The second version of H.323 was defined with a view to meeting these new requirements[7]. The scope of this version was extended to cover metropolitan area networks (MANs), wide area networks (WANs) and many others (like Internet Packet Exchange IPX).

H.323 relies on many other protocols defined by the ITU-T:

- H.225.0 for call signalling and registration, admission and status (RAS)
- H.245 for control signalling
- H.320 for integration with Integrated Services Digital Networks (ISDN)
- H.324 for integration with Switched Circuit Networks (SCN), like the PSTN

An important requirement of H.323 is that its components must interwork with multimedia communication terminals for other types of network connection. The most significant example is the connection between an IP-based network and the PSTN to provide IP telephony. This is the reason why the Q.931 access protocol has been chosen as the basis for the protocols of recommendations H225.0 and H.245. The interoperability between these networks is achieved, as specified in chapter three, through a gateway. It should also be noted that H.323 is completely

[7] Since then, version 3 and 4 were approved in the end of 1999 and 2000 respectively. However, these new versions bring new functionality's while the basic mechanism is the same.

independent from the transport layer. Whereas H.323 is used over an IP-based network, RTP is systematically used to transport the media information[8].

A5.1.1 H.323 components

According to [H323], the H.323 protocol specifies four components: terminal, gateway, gatekeeper and Multipoint Control Unit (MCU).

The H.323 terminal is an end-point used for real-time full-duplex communications. It can be either a PC or a stand-alone device that implements the H.323 stack. According to the specification, the terminal must support the G.711 audio codec. Video codec is also possible but does not fall within the scope of this dissertation.

The H.323 gateway provides connections between a H.323 network and a non-H323 network. An application of the H.323 gateway is in IP telephony, where the gateway connects an IP network with an SCN network. The H.323 gateway must provide the same basic features as the generic gateway we covers in chapter 3, i.e. translation of call setup and release, conversion of the media stream format and its transport. It is important to note that the gateway is not required for communication between two H.323 terminals on the same network.

The H.323 gatekeeper is optional in the H.323 standard. The gatekeeper provides addressing, authorization and authentication of terminals and gateways; bandwidth management, call routing services, address translation, zone management. Although they are optional, terminals and gateways must use the functionalities of the gatekeeper if there is one. In the case of IP telephony, there must be a gatekeeper in order to perform E.164 address translation into transport address.

The H.323 multipoint control unit (MCU) is used for conference between three or more terminals. All terminals that participate in a conference must establish a connection with the MCU that will manage the conference. The 4 components of an H.323 network are schematised in figure A5.1

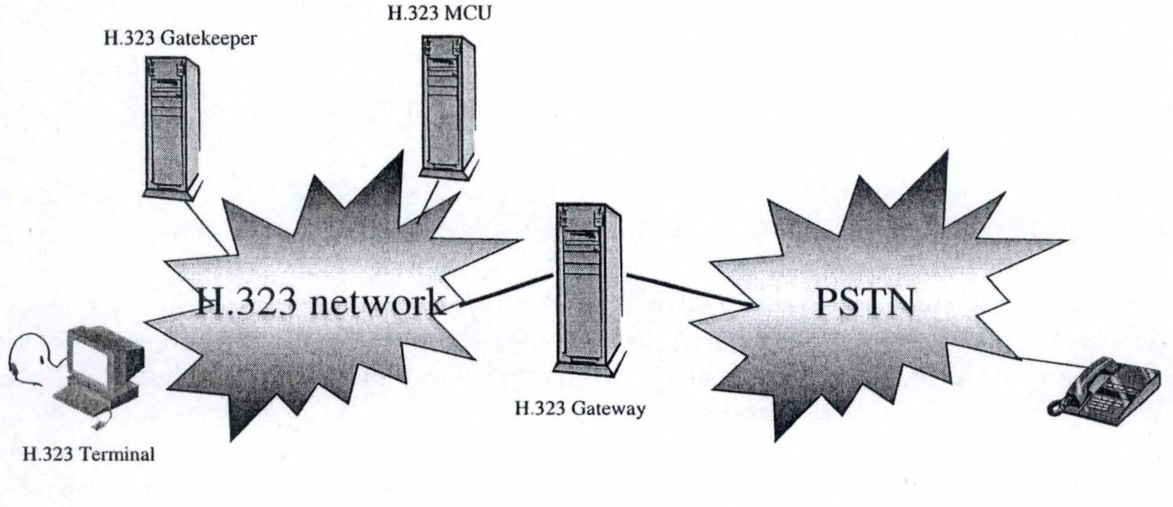


Figure A5.1 – H.323 components

[8] Such media are called, in the H.323 terminology, the media stream.

An H.323 zone is the collection of all terminals, gateways and MCUs managed by a single gatekeeper. A zone must have at least one terminal and contains one, and only one, gatekeeper. A zone may be independent of network topology and may be made up of network segments interconnected by routers, for instance.

A5.1.2 H.323 protocols

In this section, we give a short survey of H.245 (see [H245]) and H.225.0 (see [H225.0]) protocols. It is impossible to cover all the protocols that H.323 uses[9] so the reader may consult the ITU-T Recommendations for more details. H.323 draws on the constructs of calls and channels for modelling a communication session. A call is defined as point-to-point communication between H.323 endpoints (terminals and gateways). Channels are used as building blocks for calls: a collection of channels between two endpoints makes up a call. Channels are normally unidirectional and can be reliable or unreliable.

The H.225.0 protocol can be split into two functionalities: first, the registration, admission and status (RAS) and then, the call signalling. RAS is the protocol used between endpoints and gatekeepers. Of course, in network environments where there is no gatekeeper, RAS should not be used. RAS is used to perform registration, admission control, bandwidth changes, status and disengage between endpoints. It is also used to discover the gatekeeper. The registration is performed by endpoints with the aim of giving to the gatekeeper the transport addresses and alias addresses of the endpoints. Every endpoint that has registered with a gatekeeper forms part of its configuration. This registration allows the gatekeeper to locate the endpoints, i.e. the translation of an alias name or E.164 address into a transport address. To perform all these functions, an unreliable RAS channel is used to exchange RAS messages. This channel is opened between the endpoint and the gatekeeper prior to the establishment of any other channel.

The call signalling is used to set up connections between H.323 endpoints, over which the media stream can be transported. A reliable call-signalling channel is opened to exchange H.225.0 messages. This channel can be opened between endpoints when there is no gatekeeper. When a gatekeeper exists, two solutions exist. The first solution is to exchange H.225.0 messages directly between the endpoints: this is called direct call signalling. The second solution is to exchange the H.225.0 messages between the endpoints after being routed by the gatekeeper: this is called gatekeeper-routed call signalling. The gatekeeper chooses the method during RAS-admission message exchange.

The H.245 protocol is used to negotiate and establish connections between H.323 endpoints. The H.245 messages are carried over a control channel established between the endpoints, an endpoint and a MCU or an endpoint and a gatekeeper. The H.245 control channel is permanently open, unlike the media channels. The features of the H.245 protocol are:

- Capabilities exchange. For example, two endpoints negotiate the audio codec they are going to use by exchanging the audio codec they support
- Opening and closing of logical channels used to transport the media stream. A logical channel is unidirectional
- Flow-control messages

[9] And, as we shall see later, this is one of the major drawbacks of the H.323 protocol.

The MCU uses H.245 messages and procedures to establish multipoint communications between endpoints. The possibility to send H.245 messages to the gatekeeper is valuable because it allows a better control of the calls in the network.

A5.1.3 Putting these protocols together

In this section we give some examples showing how all these protocols can be combined to provide a communication between two H.323 endpoints. According again to the ITU-T Recommendation H.323, the provision of the communication is provided in the following steps:

- Call setup
- Initial communication and capabilities exchange
- Establishment of audiovisual communication
- Call services
- Call termination

First, call setup. Figure A5.2 shows how two H.323 terminals (respectively called T1 and T2 in the remainder of this section) connected to the same gatekeeper can set up a call where the gatekeeper has chosen the direct-call signalling method.

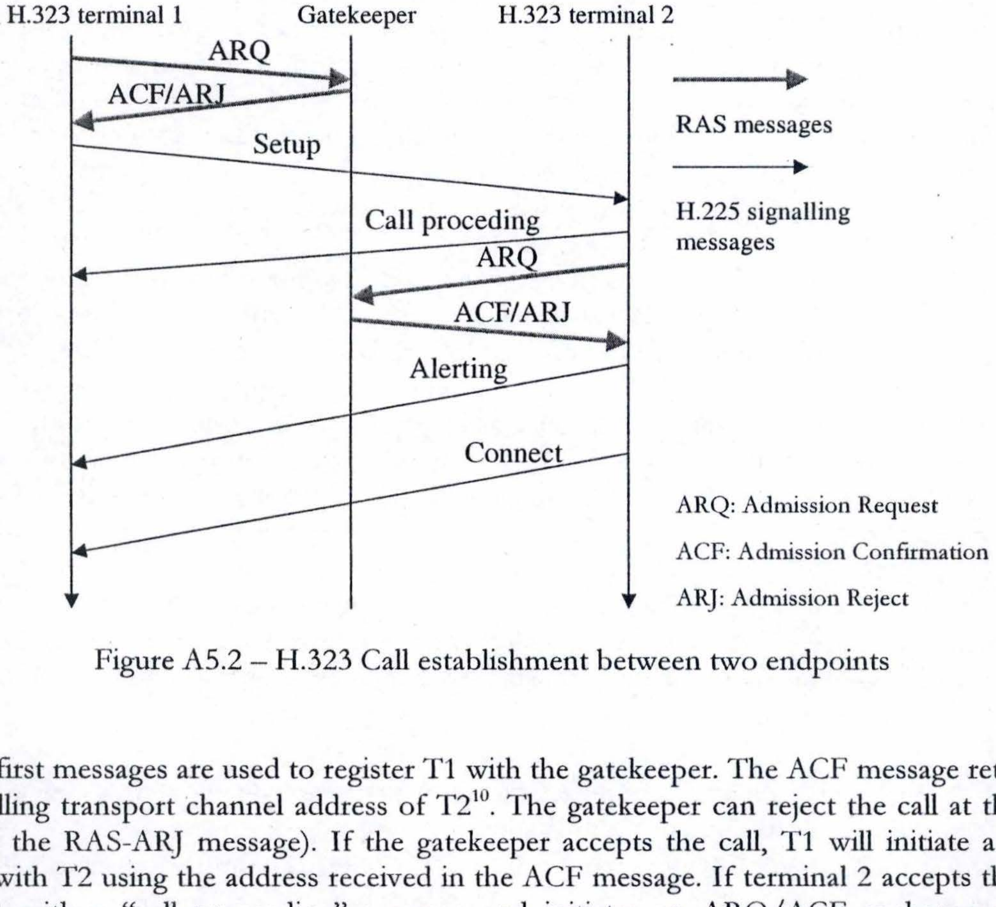


Figure A5.2 – H.323 Call establishment between two endpoints

The two first messages are used to register T1 with the gatekeeper. The ACF message returns the call signalling transport channel address of T2[10]. The gatekeeper can reject the call at this point (by using the RAS-ARJ message). If the gatekeeper accepts the call, T1 will initiate a "setup" message with T2 using the address received in the ACF message. If terminal 2 accepts the call, it warns T1 with a "call proceeding" message and initiates an ARQ/ACF exchange with the gatekeeper. Again, it is possible that the gatekeeper rejects the call, in which case the latter sends "release complete" to T1. If the call is accepted, T2 sends an alerting message to notify T1 that

[10] Or that of the gatekeeper in the case of gatekeeper-routed call signalling.

the connection has been established. T2 also sends a connect message that contains a control channel transport address for use in H.245 signalling.

This is a simple example of call establishment and there are many others such as when there is no gatekeeper, when only the calling party has a gatekeeper, etc. The reader can find all possible situations of H.323 call establishment in [H323, pp. 41-61]. Once the call has been established, the initial communication and capabilities exchange can begin. This second step is schematised in figure A5.3.

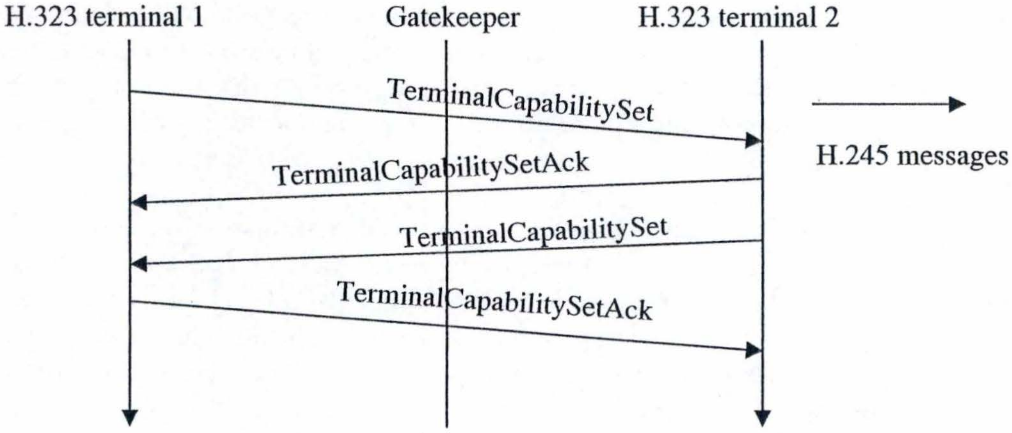


Figure A5.3 – Initial communication and capabilities exchange

In figure A5.2, T1 receives the control channel transport address. The first message to be sent using this address is a "TerminalCapabilitySet" message. This message contains the capabilities that T1 supports (e.g. audio code, voice code, etc.). T2 acknowledges T1's capabilities and sends its capabilities. T1 acknowledges T2's capabilities. It is important to note that the "TerminalCapabilitySet" message must be the first H.245 message sent. If the capability exchange is successful, the endpoints proceed to the next step, i.e. to the establishment of audiovisual communication. This step is schematised in figure A5.4 below.

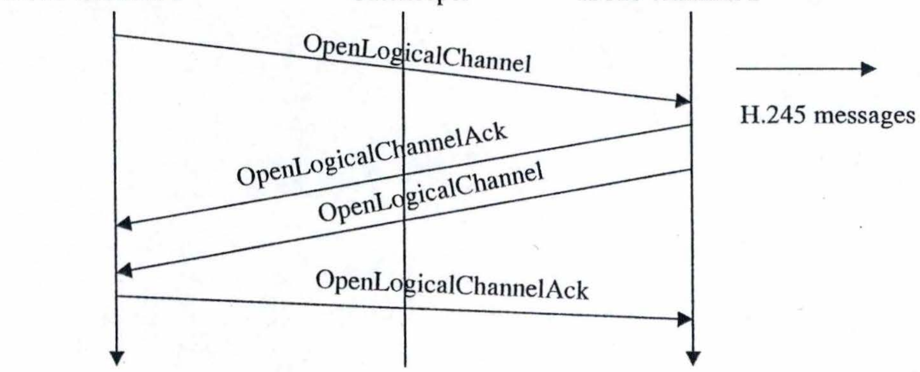


Figure A5.4 – Establishment of audiovisual communication

We assume here that the underlying H.323 network is an IP-based network The "OpenLogicalChannel" message is used to open a media channel. The message also contains the transport address of the RTCP (Real-time Transport Control Protocol) channel. The "OpenLogicalChannelAck" message is used to acknowledge the establishment of the unidirectional logical channel from T1 to T2. The message also contains the RTP transport address allocated by T2 to be used by T1 to send the media stream. Then the reverse operation is performed for T2. Of course, as foreseen in the H.323 standard, any other network can be supported but We found relevant to give a practical example of the protocols that can be used to perform the communication itself. At this point, two unidirectional channels are established between T1 and T2 to transport the media stream (e.g. voice traffic).

During the communication, some services can be provided: this is the fourth step. For example, the endpoints or the gatekeeper can request a bandwidth change (BRQ-RAS message). The gatekeeper can request an information request (IRQ-RAS message) in order to retrieve the status of any endpoint registered with it. A diagnostic device can also use this message. Many other services can be requested, especially during multicast conferencing but this is beyond the scope of this basic overview of the H.323 protocol.

The fifth and last step is the call termination. The call termination is initiated by an "EndSessionCommand" message. The other endpoint also sends an "EndSessionCommand" to the first one to acknowledge the release and to release its channels. The release is complete when the initiator endpoint sends a "release complete" message to the other endpoint. As we have seen, the gatekeeper performs bandwidth management for its zone and then it is important that the endpoint disengages with its gatekeeper when a call is released. The call termination is schematised in figure A5.5 below[11].

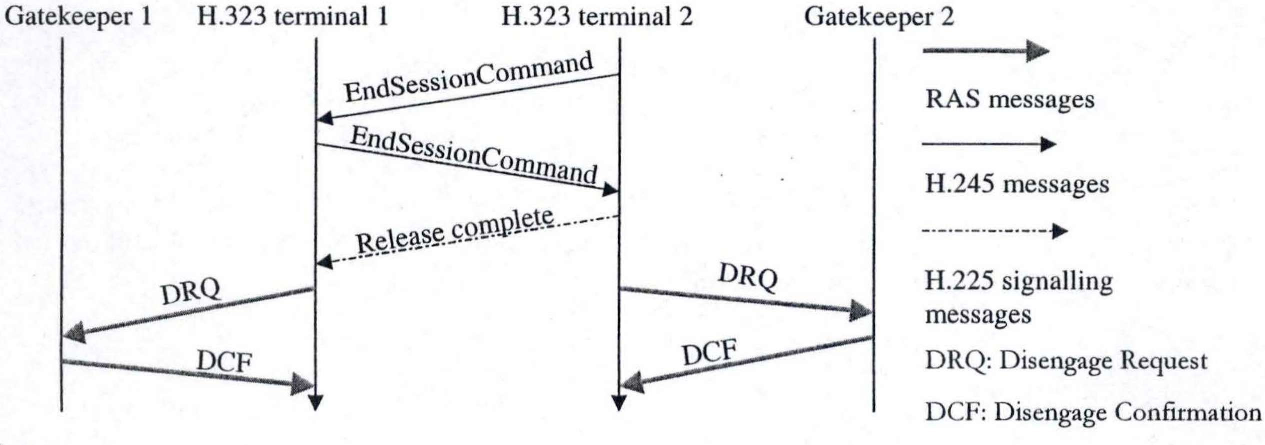


Figure A5.5 – Call termination

A5.1.4 H.323: Conclusion

Without anticipating the last section of this appendix, We admit that the H.323 protocol has not convinced us: it is very hard to understand and they are many protocols associated with it. Moreover, H.323 was not designed initially for Internet telephony services. The decision of the ITU to be backward compatible has fundamentally increased the complexity of this protocol.

[11] In this example, T1 and T2 have different gatekeepers. The aim is to show another situation between H.323 endpoints. There is no link with the three last figures even if they are supposed to model the same call.

However, H.323 has an excellent interoperability with switched circuit networks like the PSTN. More details are given in the last section of this appendix.

A5.2 Session Initiation Protocol

Session Initiation Protocol (SIP) is a client-server protocol, similar to HTTP, used to create, modify and terminate sessions with one or more participants. Like a browser making requests to a web server, the SIP client generates requests to a receiving entity (a server) that processes them and then sends back responses. Even if SIP is similar in both syntax and semantics to HTTP, it provides its own methods and headers for providing the functions required in IP telephony signalling. In this section, we cover the features of SIP as well as the description of its protocol as defined in [RFC2543].

A5.2.1 Definition

[RFC2543] defines Session Initiation Protocol this way: "The Session Initiation Protocol (SIP) is an application-layer control (signalling) protocol for creating, modifying and terminating sessions with one or more participants. These sessions include Internet multimedia conferences, Internet telephone calls and multimedia distribution. Members in a session can communicate via multicast or via a mesh of unicast relations, or a combination of these. SIP invitations used to create sessions carry session descriptions, which allow participants to agree on a set of compatible media types. SIP supports user mobility by proxying and redirecting requests to the user's current location. Users can register their current location. SIP is not tied to any particular conference control protocol. SIP is designed to be independent of the lower-layer transport protocol and can be extended with additional capabilities." All the concepts in this definition are fully described in the remainder of this section. However, the concept of session must first be defined: "a multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session" ([RFC2543, p. 10]).

SIP forms part of the IETF conference control signalling, which consists of three other main protocols. The first, Session Description Protocol (SDP) is generally used by SIP to transport the session description. The second, the Real Time Streaming Protocol (RTSP), is an application-layer protocol for control over the delivery of data with real-time properties (see [RFC2326]). The last one, the Session Announcement Protocol (SAP, see [SAP96]), can be used to announce multimedia sessions. SAP can, like SIP, use SDP to describe the session. The latter is described in the next section.

A single call may involve several servers and clients, as requests[12] may be forwarded. A call participant may either generate or receive requests. Thus SIP-enabled end systems include a protocol client and server (respectively called a user agent client and a user agent server). A call participant can invite both persons and "robots". The initiator, i.e. the entity that initiates the session, does not have to be a member of the session to which it is sending out an invitation. For example, a proxy may establish a call between two users (see figure A5.6). Media and participants can be added to an existing session.

[12] The request and the responses which follow represent an SIP transaction.

A5.2.2 Session Description Protocol

We give here a short survey of the Session Description Protocol (SDP) as defined in [RFC2327]. The Session Description Protocol is a protocol intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation. SDP can be the payload of SIP messages but other protocols, such as SAP, use it[13]. SDP provides a means to convey sufficient information to allow joining and participating in the session. SDP payload includes:

- Session name and purpose
- Time(s) the session is active
- The media comprising the session
- Information to receive those media (addresses, ports, formats and so on)

As resources necessary to participate in a session may be limited, some additional information may also be desirable:

- Information about the bandwidth to be used by the conference
- Contact information for the person responsible for the session

Sessions may be bounded or unbounded in time. Whether or not they are bounded, they may be only active at specific times. SDP can convey timing information for the purpose of time zone adjustments, for example. The media information is very complete. It describes the type of media (audio for example), the transport protocol (RTP over UDP over IP), the format of the media (MPEG video), the IP address to contact, the transport port to contact and finally media specific information (for bandwidth description, for example).

The SDP content is composed of <FIELDNAME>=<FIELDVALUE> lines separated by CRLF. SDP content is structured into three parts: first, the session description fields describe the session in general. Then, the time description fields describe timing specific information. Finally, the media description fields define the media which is conveyed on this session. The following two tables (tables A5.1 and A5.2) show the last part of an SDP content, that is time description fields and media description fields. An asterisk indicates optional fields.

Field Name	Description
t	Time the session is active
r *	One or more repeat times

Table A5.1 – Time description fields

Field Name	Description
m	Media name and transport address
i *	Media title
c *	Connection information[14]
b *	Bandwidth information

[13] If SDP is used as the payload of SAP, it is typically used to announce multicast sessions.

[14] This field is optional if it is included at session-level.

k *	Encryption key
a *	Zero or more media attributes lines

Table A5.2 – Media description fields

Table A5.3 shows the main structure an SDP payload may contain.

Field Name	Description
v	Protocol version
o	Owner/creator and session identifier
s	Session name
i *	Session information
u *	URI[15] of description
e *	E-mail address
p *	Phone number
c *	Connection information[16]
b *	Bandwidth information
One or more time descriptions (see table A5.1)	
z *	Zone time adjustments
k *	Encryption key
a *	Zero or more session attributes line
Zero or more media descriptions (see table A5.2)	

Table A5.3 – Structure of an SDP message

Each of these fields is fully described in [RFC2327]. This section ends with an example of SDP content:

```
v=0
o=snicoll 584486 464564 IN IP4 138.48.32.107
s=SIP transaction using SDP example
u=http://www.info.fundp.ac.be/~snicoll/sip.ps
e=snicoll@info.fundp.ac.be (Stéphane Nicoll)
c=IN IP4 sip.info.fundp.ac.be
m=audio 5000/2 RTP/AVP 0
```

The "o" field is composed of six sub-fields: first, the user's login on the originating host (snicoll). Secondly, the session identifier (584486). Thirdly, the version number for this announcement (as said before, changes can be made during a session and proxies must identify these changes). Fourthly, the network type (IN for Internet and not for Intelligent Network!). Fifthly, the address type (here IP4 stands for Internet Protocol version 4) and finally the globally unique address of the machine from which the session was created. A description of the session can be found at http://www.info.fundp.ac.be/~snicoll/sip.ps. The "c" field has the same semantics as the "o" field (network type, address type and connection address). The "m" field describes the media that

[15] Uniform Resource Identifier.

[16] This field is not required if included in all media description.

will be carried in this session. It is composed of four sub-fields. The first, the media type, defines the type of media (here audio but other types are defined, like video, application, etc.). The second is the transport port(s) to use. There is a possibility to specify multiple transport ports. In the example, 5000/2 means that port 5000 and 5001 will be used for the first RTP/RTCP pair while the second one takes ports 5002 and 5003. The third sub-field is used to specify the transport protocol. The transport protocol values are dependent on the address-type field in the "c" fields. Thus a "c" field of IP4 means that the transport protocol runs over IPv4. For IPv4, it is normally expected that most media traffic will be carried as RTP over UDP. In the example it is RTP using the Audio/Video profile (AVP, see [RFC1890]) carried over UDP. The sub-field is media formats. For audio and video, this will normally be a media payload type, as defined in the RTP Audio/Video Profile. In the example, 0 is used to describe μ-law PCM coded single channel audio sampled at 8KHz type. Of course, this is a simple example but the study of SDP is not a critical issue for the scope of this dissertation.

A5.2.3 SIP methods and messages

There are two kinds of SIP messages: requests and responses. The SIP messages are, like HTTP, textual-encoded and contain header fields, which define call properties and service information. A SIP request begins with a request line and is followed by headers. The request line is composed of the SIP method token, the requested URI and the SIP protocol version. An SIP response begins with a status line and is also followed by headers. The status line is composed of the SIP protocol version, the numeric status code and a textual phrase associated with the status code. Status codes are similar to HTTP and are classified in 6 classes. Responses of the 1xx class indicate call progress and they are always followed by other responses indicating the final outcome of the request. 2xx class indicates success, 3xx redirection. Finally 4xx, 5xx and 6xx classes are used to notify client, server and global failures respectively.

The SIP headers may be divided into four different groups, according to [RFC2543]:

- General header fields applied to both request and response messages
- Entity header fields define information about the message body or, if no body is present, about the resources identified by the request
- Request header fields act as request modifiers and allow the client to pass additional information about the request and the client itself to the server
- Response header fields allow the server to pass additional information about the response which cannot be placed in the Status-Line

SIP reuses many of the header fields used in HTTP, such as the entity and authentication headers. This facilitates the integration of SIP servers with web servers and allows for code reuse. The logical call source indicates the entity that is requesting the call (the initiator); the logical call source is conveyed in the "Form" header field. The logical call destination contained in the "To" field names the party whom the originator wishes to contact (the recipient). The media destination conveyed in the payload of SIP messages has already been covered. However, it is important to point out that SIP can convey any MIME[17] type as payload. Each call is logically identified by a globally (in time and space) unique call identifier (carried in the "Call-ID" field). The call identifier is created by the originator and used by all call participants.

[17] Multipurpose Internet Mail Extensions (see [RFC2231]).

The SIP protocol defines six main methods. INVITE is used to invite a user to a call; OPTIONS asks for capabilities exchanges (but does not set a connection); BYE terminates a connection between two users in a call; REGISTER contains information about a user's location to an SIP server; ACK is used for reliable message exchanges and CANCEL terminates a search for a user. Moreover, additional SIP methods can be added easily. An example of such an extension is the SIP INFO method defined in [RFC2976]. The purpose of the INFO method is to allow the carrying of session related control information that is generated during a session (e.g. ISUP).

SIP makes minimal assumptions about the underlying transport protocol. However, there is the restriction that a whole SIP request or response has to be either delivered in full or not at all. An important reliability feature is that SIP does not tie a session to the existence of a TCP connection (in the case of an Internet environment), allowing to be maintained calls even across reboots of some of the participants, as long as the end systems maintain call identifiers.

For addressing, SIP makes use of uniform resource identifiers (URI's), which are generalisations of Uniform Resource Locators (URL's) in common usage on the web. SIP defines its own URI and can carry other URI's, such as HTTP, etc. SIP chose e-mail like identifier for the called party of the pattern name@domain or phone-number@gateway. E-mail addresses offer a basic location-independent form of addressing, in that the address does not have to designate a particular Internet host, but can be a domain, which is resolved into one or more possible domain mail server hosts. For e-mail, finding the mail exchange host is often sufficient to deliver mail (as the user uses protocols such as POP to retrieve the mail). However, for real-time communication, data are exchanged between the workstations of the end-users. Thus, SIP must be able to resolve name@domain into user@host. The SIP location server, for example, fulfils this role.

We have already seen that SIP reuses many well-known protocols of the Internet. Domain Name System (DNS) and Light Directory Access Protocol (LDAP) are no exceptions. DNS is most often used to map an URL to an IP address (typically the mapping of a web site to its IP address). This mapping is made by hierarchical requests. This is the reason why DNS is not suited for looking up individual subscribers (DNS is not used to look up individual e-mail addresses anyway). However, DNS can be used by SIP to map the name of a domain to the name of a server for a specific request. LDAP is a true directory service with search of attributes to a record. Such a database is appropriate to obtain a globally unique user handle from a set of attributes. SIP servers can make use of LDAP servers to look up matches for "Pierre Dupont" or "Dupont" for example. SIP can also use a programmable directory service[18], where the response may depend on any factors like the time of the day, user's status, the captain's age or whatever else.

Before reaching the final user, the SIP request can traverse many proxy servers (especially when this user has used the user mobility feature). As a result, proxy servers are primarily responsible for call routing. This route may be determined by many factors, like database query, for example. A server may also act as a redirect server, i.e. informing the client of the address of the next hop server with the aim that the client contacts it directly instead of proxying the request.

[18] This is one example of IN services provided on a packet network like the Internet. Many more examples and architectures are presented in chapter four.

A5.2.4 SIP in practice

The most important operation is to invite a participant in a call. First of all, the originator must obtain the address where the user is to be called[19] (of the generic form name@domain). The user tries to translate it and once he has found it, he sends an INVITE message. In order that this invitation succeeds, Alice, who is using the crow.example.com machine must have first registered with her local SIP server (example.com). This is done via the following SIP (request) message.

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP crow.example.com
From: sip:alice@example.com
To: sip:alice@example.com
Call-ID: 70710@crow.example.com
CSeq: 1 REGISTER
Contact: <sip:ali@crow.example.com:5000;transport=udp>
Expires: 14400
```

The registration expires after four hours. Any future invitations to alice@example.com arriving at sip.example.com will now be redirected to ali@crow.example.com, UDP port 5000. Figure A5.6 shows how Bob tries to call Alice in proxy server mode.

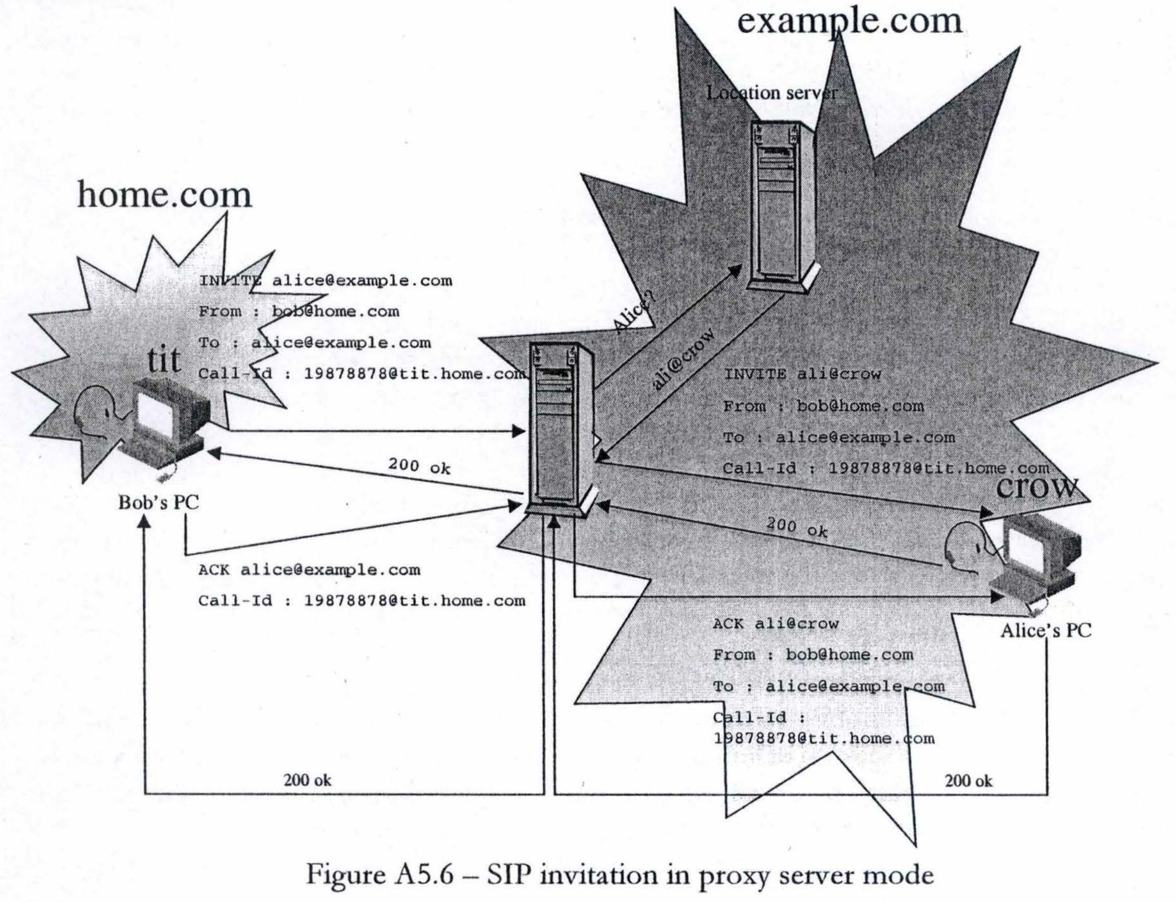


Figure A5.6 – SIP invitation in proxy server mode

Another feature of SIP is the user mobility feature. If Alice wants to be reached elsewhere, for example at her desk, she updates her reservation after first cancelling any existing locations. The

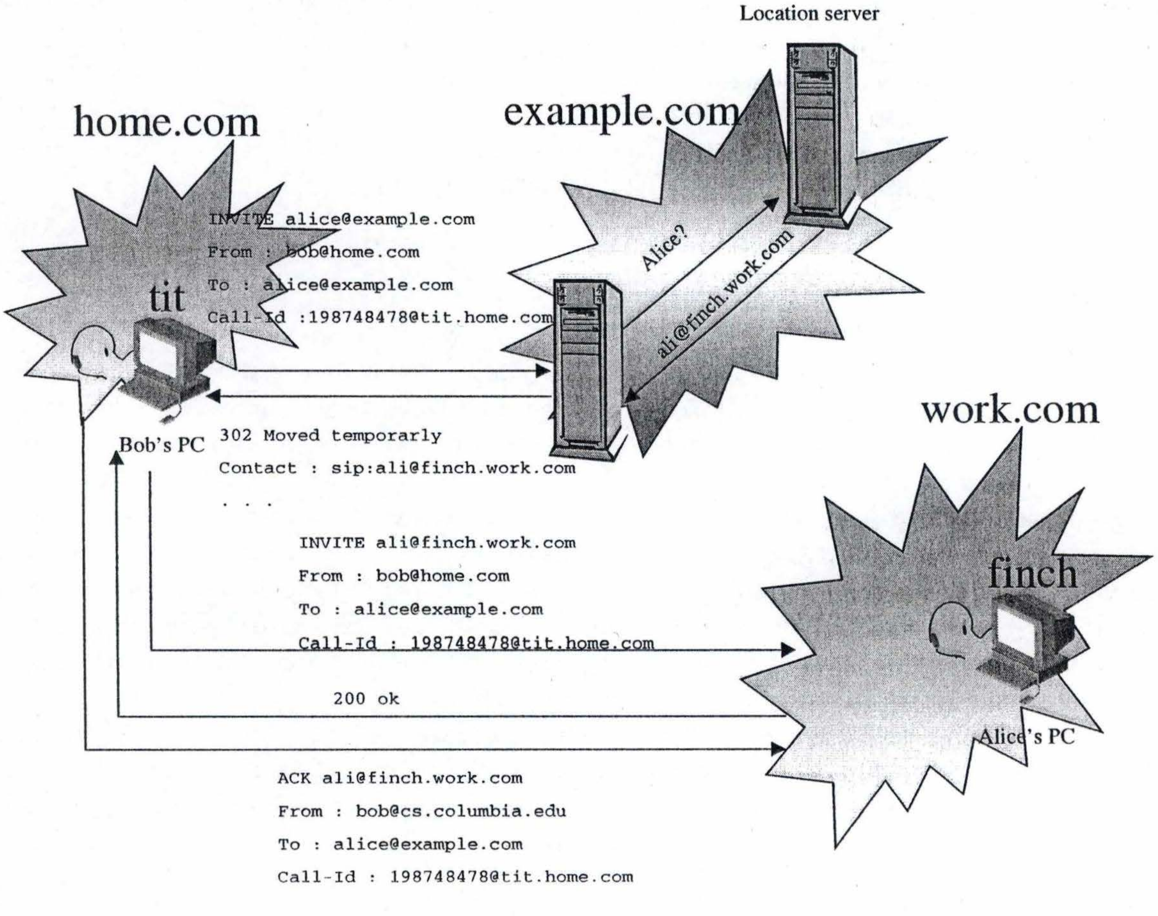
[19] If the originator knows this address, he can contact the user directly by sending an "invite" message.

first message cancels any existing locations while the second one makes a new reservation. Note that the call identifier is always the same.

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP crow.example.com
From: sip:alice@example.com
To: sip:alice@example.com
Call-ID: 70710@crow.example.com
CSeq: 2 REGISTER
Contact: *
Expires: 0

REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP crow.example.com
From: sip:alice@example.com
To: sip:alice@example.com
Call-ID: 70710@crow.example.com
CSeq: 3 REGISTER
Contact: sip:alice@work.com
```

Now the server will forward any request for Alice to the server at work.com, using the Request-URI alice@work.com. For the server at work.com to reach Alice, she will need to send a REGISTER there or inform the server of her current location through some other means. Figure A5.7 shows what happens when Bob tries to call Alice again when she is at work (this is the SIP invitation in redirect mode).



```
INVITE alice@example.com
From : bob@home.com
To : alice@example.com
Call-Id :198748478@tit.home.com

302 Moved temporarly
Contact : sip:ali@finch.work.com
. . .

INVITE ali@finch.work.com
From : bob@home.com
To : alice@example.com
Call-Id : 198748478@tit.home.com

200 ok

ACK ali@finch.work.com
From : bob@cs.columbia.edu
To : alice@example.com
Call-Id : 198748478@tit.home.com
```

Figure A5.7 – SIP Invitation in redirect mode

It should be noted that SIP messages on the last message are not fully correct since they do not provide the full SIP message content. This choice has been made to improve the readability of the figures. We end this section by providing a SIP response message example. The following SIP response message of Figure A5.7 above notifies Bob that Alice has moved temporarily to her desk.

```
SIP/2.0 302 Moved temporarily
Via: SIP/2.0/UDP tit.home.com
From: bob@home.com
To: alice@example.com
Call-ID: 198748478@tit.home.com
Contact: sip:ali@finch.work.com
CSeq: 1 INVITE
```

A5.2.5 SIP: Conclusion

SIP is really an interesting protocol. Its interaction with IP applications such as the web and e-mail is a powerful advantage compared to H.323. This protocol certainly merits a dissertation in itself but, unfortunately, we have not enough time to provide more details.

A5.3 A comparison between SIP and H.323

We have seen two protocols that are resolving the same problem namely providing a signalling standard for Internet telephony. Now we must choose one of them because the best standard that aims to improve the convergence must above all be unique and accepted worldwide. Thus in the remainder of this dissertation, we will restrict ourselves to the protocol that we are going to choose now. This comparison is sub-divided into the important properties a good protocol must have. These are, for example, simplicity, scalability, extensibility, etc.

A5.3.1 Simplicity

Nobody will deny that simplicity is a critical issue for a protocol that wishes to be accepted worldwide and at least for two main reasons. First of all, a lower complexity allows the implementation of the protocol in a device where resources are limited (for example, mini-notebooks). Then, a lower complexity limits the interpretation of the protocol, i.e. different implementations of the same mechanism due to a lack of precision. H.323 is an extremely complex protocol. Backbiters point out that the number of pages between the H.323 protocol suite and the SIP protocol suite is rather disproportionate (130 pages for SIP and more than 800 pages for H.323). We have seen in the preceding section that SIP reuses many fields of HTTP and is text-encoded. This allows existing HTTP parsers to be quickly modified for SIP usage. H.323 uses ASN.1-based message representation. This choice involves greater complexity and more implementation issues (a specific ASN.1 parser for example). Moreover, the SIP textual encoding allows faster debugging, manual entries, etc. The absence of clean separation between the different protocols that H.323 uses is another proof of its complexity. For example, call forward uses three different protocols to complete (H.450[20], H.225.0 and H.245). Also, a single

[20] The ITU-T Recomendation H.450 defines a generical functional protocol for the support of supplementary services in H.323 (see [H.450.1]).

task may be accomplished in different ways. For example, there are three distinct ways in which H.245 and H.225.0 may be used together: we have seen the original approach of separate connections in Figures A5.2 and A5.3. But there are also two another ways known as "fast start" and "H.245 tunnelling through H.225.0". Finally, some features of the H.323 protocol suite are already defined in another sub-protocol. This is the case with the feedback and conference control features of H.245 that are already present in RTCP. This redundancy is unacceptable in an environment where reusability is a critical concern. Unlike H.323, SIP is an extremely simple protocol. With its six methods and its few headers, the clear purpose of the SIP standard is to provide a strong basis of signalling, allowing extension to be developed as easily as possible. We have seen an example of this with the INFO method[21]. These extensions are really annoying since many specialists are creating their own extensions and finally there is no real version of SIP. This problem is so acute that an Internet Draft has been written for these specialists (see[RSGL01]). The purpose of this draft is to propose guidelines to avoid a clash between all these extensions.

A5.3.2 Extensibility

SIP has learned from the experience of the HTTP protocol: by default, unknown headers and values are simply ignored (see the Require header); the feature names are based on a hierarchical namespace, and new features names can be registered with IANA[22]. Furthermore, the organization of error codes allows the adding of new codes to a class, while achieving compatibility. H.323 provides extensibility mechanisms as well: they are announced by the "nonStandardParam" fields placed in various locations in the ASN.1. This field is structured in a vendor code followed by a value that has a meaning only for this vendor. However, there is no way of adding a new value to some existing parameter if not explicitly foreseen. Another extensibility issue is audio and video codecs. Codecs are identified by string names which can be registered with IANA. SIP uses SDP to convey the codecs supported by an endpoint in a session. H.323, however, requires that each codec is registered and standardised, i.e. each codec must have an ITU codepoint. This is an important limitation for small players that may use free codecs. Moreover, SIP defines a simple third-party call control that allows one third party to instruct another entity to create and destroy calls to other entities. H.323 does not provide a generic third-party control mechanism. Another example that shows the theoretical superiority of SIP is its modularity: SIP can be used to locate a user and specify an H.323 URL, indicating that the actual communication should take place with H.323, with no change to SIP at all.

A5.3.3 Scalability

An important problem reported by many specialists is the absence of a loop detection algorithm in H.323[23]. SIP has, however, a loop detection algorithm similar to the one used in BGP (Border Gateway Protocol) which can be used stateless. Then, SIP transaction can be completely stateless, i.e. servers receive a call request, process it and then completely forget about it. SIP messages contain sufficient state to allow for the responses to be forwarded correctly. In H.323, gatekeepers that are on the local loop must be stateful: they must keep call state for the entire duration of the call. Thus, as gatekeepers use TCP, they must hold TCP connections for the

[21] Another important extension is the PSTN/Internet Internetworking (PINT) that relies on SIP and SDP. PINT is discussed in chapter four.

[22] Internet Assigned Number Authority, http://www.iana.org.

[23] It can be done state fully by storing messages.

entire duration of the call. SIP however can be carried on either with TCP or UDP and, in the case of UDP, no connection state is required. The use of UDP is not a problem since SIP provides its own reliability mechanism. Moreover, the use of UDP in place of TCP has many advantages, such as the possibility of setting up the retransmission timer.

A5.3.4 Interoperability

Interoperability is the strongest advantage of H.323. It brings with it a host of well known support protocols upon which it is based (e.g. the Q.93x series of protocols), which is a good reuse of building blocks. At the time of writing, SIP cannot compete with the H.323 interoperability and thus all solutions involving a PSTN – IP integration are using the H.323 protocol suite. However, a great number of initiatives are emerging to improve the interoperability of SIP with switched circuit network (see [SIP-T00]).

A5.3.5 H.323 or SIP?

A considerable amount of work is currently being done at the IETF regarding SIP. In fact, SIP is really a fashion protocol and we think it could be a revolution in terms of signalling on the Internet. Most probably the reader has learned much more from SIP than from H.323 while they have been allocated roughly the same number of pages in this appendix. H.323 was originally designed as a standard for videoconferences over LANs. It subsequently evolved to meet the market in terms of Internet telephony but the overall feeling is that this evolution has been speeded up by market needs. This would account for this huge complexity (accentuated by the need to maintain compatibility with other ITU protocols). This comparison of SIP and H.323 is not exhaustive: security issues like security between H.323 gatekeepers have not been tackled. The reader is invited to consult [SR98] and [PW99], for example. Four points remain to be dealt with. The first is that, even if SIP implementations have already been proposed by vendors, no raw performance comparison between H.323 and SIP has yet been published. One must verify that the theory matches the real performance. The second point is that H.323 has a better interoperability with the PSTN than SIP and thus efforts must be made in order to improve the SIP interoperability. The third is that SIP and H.323 can be combined in the same network. For example, SIP can be used to locate the called party and the actual communication can be made using the H.323 protocol. And the last one is protocol evolution. New versions of H.323 (especially H.323v4) seems to resolve lots of problems and, in mid-term, H.323 and SIP will probably coexist.