

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

NOTE TO USERS

This reproduction is the best copy available.

UMI[®]

Light-weight SIP protocol for Internet Telephony services

Evelina Evloguieva

**School of Computer Science
McGill University
Montreal, Quebec
July 1999**

**A thesis submitted to the Faculty of Graduate Studies and Research in
partial fulfillment of the requirements for the degree of Master of Science**

©Evelina Evloguieva, 1999



National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file *Votre référence*

Our file *Notre référence*

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-55052-4

Canada

Abstract

The technology that governs the Telecommunications today is based on the Intelligent Networks (IN). But we can see that the Internet Telephony is emerging rapidly and that it has good chances to become the basis for the next generation telecommunication networks. There are two major competing standards for Internet Telephony – H.323 protocol stack and SIP. This thesis focuses on SIP and the Value Added Services in the SIP based Internet Telephony. It describes and analyzes two existing SIP based approaches to the VAS implementation and presents new hybrid SIP-IN approach based on the concept of reusing the existing IN nodes. The major part of the study is devoted to the design of a lightweight protocol, built as SIP extension, providing for VAS in the hybrid SIP-IN environment. To illustrate the hybrid SIP-IN approach to VAS implementation and the SIPext protocol operation the execution of the Freephone and the Call Distribution services is described. Finally the functional modules supporting the SIPext communication in a hybrid SIP-IN architecture are outlined.

Résumé

Les Réseaux Intelligents (Intelligent Networks - IN) constituent aujourd'hui une technologie de pointe dans le domaine des télécommunications. Cependant, la téléphonie basée sur les technologies de l'Internet progresse à une telle vitesse qu'il est très probable qu'elle constitue la technologie de base pour la prochaine génération des réseaux de télécommunication. Il existe aujourd'hui deux principaux standards pour la téléphonie IP – H.323 (ITU-T) et SIP (IETF). Cette se concentre sur SIP et sur les services à valeur ajoutée dans un réseau de téléphonie IP basé sur SIP. Elle décrit et analyse deux approches proposées actuellement pour implémenter des services à valeur ajoutée dans un réseau SIP, et propose une nouvelle architecture hybride SIP-IN, permettant de réutiliser des noeuds IN légèrement modifiés. La majeure partie de cette étude est dédiée à la conception d'un protocole simple, construit comme une extension de SIP, pour accéder à des services à valeur ajoutée dans un environnement hybride SIP-IN. Pour illustrer l'approche hybride SIP-IN et le protocole associé, appelé SIPext, deux études de cas sont présentées: Freephone (Appels gratuits) et la Distribution d'appels. Enfin, les modules fonctionnels supportant la communication par SIPext dans un réseau SIP-IN sont présentés.

Acknowledgments

I would like to thank my supervisors, Prof. P. Dini, R. Glitho, and Prof. G. Ratzer for giving me the opportunity of pursue this work and directing my efforts to complete it. Thank you for all your guidance and advice. I also would like to thank C. Gourraud for his responsiveness and timely assistance.

I am grateful for having the Fond FCAR financially supporting me throughout my studies and Ericsson Research Canada providing me with a cozy and friendly working environment.

Lastly, I would like to say thank you to all my family for their love and devotion in helping me through these years.

Contents

Introduction	1
1 Advanced services in classical telephony	4
1.1 Evolution of the classical telecommunication networks	4
1.2 IN platform.	6
1.2.1 Basic elements of the Intelligent Network platform	6
1.2.2 Advantages of IN	9
1.3 Semantics of the Advanced Telecommunication Services commonly offered in IN.	9
1.3.1 Number translation services.	10
1.3.2 Screening services.	11
1.3.3 Billing services	12
1.3.4 Other services	13
1.4 Semantics of the Telecommunication services offered by CMS 8800	15
1.4.1 Subscriber services	16
1.4.2 Network services	18
1.5 Conclusion	18
2 Internet Telephony	19
2.1 Overview.	19
2.2 Protocols specific to Internet Telephony	21
2.3 Signaling protocols	22
2.3.1 H.323 protocol	22
2.3.2 SIP.	23
2.4 Conclusion	25

3	Session Initiation Protocol	26
3.1	Architectural components	26
3.2	Underlying transport protocols	28
3.3	Addressing and naming.	29
3.4	SIP messages	30
3.4.1	Requests	30
3.4.2	Responses	32
3.4.3	Headers.	32
3.4.4	Message body	34
3.5	Basic operation	35
3.6	Conclusion	37
4	Value Added Services implementation using SIP	38
4.1	Service construction	38
4.2	Service logic creation.	40
4.2.1	Standard compliant service logic creation.	41
4.2.2	Proprietary service logic creation	43
4.3	Network Scenarios	43
4.3.1	Potential scenarios	44
4.3.2	Scenario analysis.	45
4.4	Building Hunt Group service in SIP environment	46
4.4.1	Hunt Group service semantics	47
4.4.2	Implementation issues	48
4.5	Conclusion.	52
5	Value Added Services implementation using hybrid SIP-IN/WIN approach	53
5.1	Network scenario.	53
5.1.1	Intelligent Network Application Protocol (INAP)	55
5.1.2	INAP environment.	56
5.1.3	INAP vocabulary	56

5.1.4	INAP architecture	58
5.1.5	INAP procedures.	59
5.1.6	INAP encoding.	59
5.2	Conclusion	59
6	Lightweight SIP protocol SCP - SIP for server communication	61
6.1	SCP – SIP server interface.	61
6.2	SIP extension – inap.sip	62
6.2.1	Architecture entities.	63
6.2.2	SIPext basic operation	66
6.2.3	Definition of inap.sip.	68
6.2.3.1	Operation: header field	69
6.2.3.2	Result-Op: header field	73
6.2.3.3	Error-Op: header field	74
6.2.3.4	Oseq: header field	74
6.2.4	Behavior of the SIPext SCP.	75
6.2.4.1	SIPext SCP acting as a server	75
6.2.4.2	SIPext SCP acting as a client	77
6.2.5	Behavior of the SIPext SSP	78
6.2.5.1	SIPext SSP acting as a client.	78
6.2.5.2	SIPext SSP acting as a server	82
6.3	Examples of IN VAS implemented in Internet Telephony with SIPext. . .	82
6.3.1	Freephone service	82
6.3.2	Call Distribution service.	87
6.4	Conclusion	93
7	Implementation issues	94
8	Conclusion and future work	101
	Bibliography	103

List of Figures

1.1	IN-based Freephone service implementation.	6
2.1	Call connections in Internet Telephony	20
2.2	Internet Telephony protocol stack	21
3.1	Architecture components of a SIP system	26
3.2	Architecture components of a SIP system connected to other networks.	28
3.3	SIP message elements	30
3.4	Request sent to a locally configured SIP server	35
3.5	Request sent to the IP address and port of the recipient's server.	36
3.6	SIP server operation for user location	37
4.1	Services reside on the end user systems	44
4.2	Services reside on a SIP server.	44
4.3	Services reside on specialized network nodes	45
4.4	Hunt Group service example.	48
4.5	Hunt Group service logic executed when a call to the subscriber B is set up . . .	50
4.6	Hunt Group service logic executed when the connection with the subscriber B is terminated	51
5.1	SIP-IN/WIN network scenario.	54
5.2	INAP – communication interface between IN nodes	55
5.3	Structure of the SS7 stack.	56
5.4	Structure of INAP	58
5.5	INAP protocol architecture.	58
6.1	SIP/IN network configuration	63
6.2	SIPext basic operation	67
6.3	Argument structure of the InitialDp operation	70
6.4	Argument structure of the CallInformationReport operation.	70
6.5	SIPext SCP acting as a server.	77
6.6	The SIPext SCP acting as a client (sending a REGISTER request)	77
6.7	SIPext SSP notifies the SCP about event detection.	80

6.8	Unsuccessful operation execution	81
6.9	User A calls Freephone number 800 3424	84
6.10	User C accepts the Freephone call.	85
6.11	The Freephone call is setup.	86
6.12	User A terminates the Freephone call	87
6.13	A call attempt to user B, who has answered less than 50 calls for the day	90
6.14	The SIPext SSP detects and reports that user B has answered a call	91
6.15	The SIPext SSP detects and reports that user B is busy	91
6.16	User B does not answer	92
6.17	A call attempt to user B, who has already answered 50 calls for the day	93
7.1	Functional modules of the SIPext SSP and SIPext SCP.	95


List of Tables

1.1	Commercial IN/WIN service packages.	14
2.1	H.323/SIP comparison	24
3.1	Response classes	32
3.2	Examples of general header fields	33
3.3	Examples of Entity header fields	33
3.4	Examples of Request header fields	33
3.5	Examples of Response header field.	34
6.1	INAP errors transport.	76

List of Abbreviations

AC	Application Context
AMPS	Advanced Mobile Phone Service
ASE	Application Service Element
ASN.1	Abstract Syntax Notation 1
BER	Basic Encoding Rules
BGP	Border Gateway Protocol
B-ISDN	Broadband Integrated Services Digital Network
BNF	Backus-Naur Form
CCF	Call Control Function
CDPD	Cellular Digital Packet Data
CMS 8800	Ericsson's Cellular Mobile System
CORBA	Common Object Request Broker Architecture
CPL	Call Processing Language
CS1	Capability Set 1
CS2	Capability Set 2
D-AMPS	Digital Advanced Mobile Phone Service
DNS	Domain Name Service
GSTN	General Switched Telephone Network
IETF	Internet Engineering Task Force
IN	Intelligent Network
INAP	Intelligent Network Application Protocol
IP	Intelligent Peripheral
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ITG	Internet Telephony Gateway
ITU	International Telecommunication Union

MACF	Multiple Association Control Function
MMUSIC	Multiparty Multimedia Session Control
MTP	Message Transfer Part
PCM	Pulse Code Modulation
PER	Packet Encoding Rules
POTS	Plain Old Telephone System
PTN	Personal Telecommunication Number
rfc	Request for comments
ROSE	Remote Operation Service Element
RMI	Remote Method Invocation
RSVP	Resource Reservation Setup Protocol
RTCP	Real Time Streaming Protocol
RTP	Real Time Transport Protocol
SACF	Single Association Control Function
SCCP	Signaling Connection and Control Part
SCF	Service Control Function
SCP	Service Control Point
SDP	Service Data Point
SIP	Session Initiation Protocol
SIPext	SIP extended with the inap.sip extension
SLP	Service Logic Program
SMS	Service Management System
SPC	Stored Program Control
SRF	Specialized Resource Function
SS7	Signaling System N7
SSF	Service Switching Function
SSP	Service Switching Point
STP	Signaling Transfer Point
TCAP	Transaction Capabilities Application Part
TDMA	Time Division Multiple Access
UPT	Universal Personal telecommunications



URL	Uniform Resource Locator
VAS	Value Added Services
WASRV	Wide Area Service Location Server
WIN	Wireless Intelligent Network

Introduction

If we would like to generalize the subject area of this work with one only word it would be “Telecommunications”. Telecommunications is a key industry in modern society. It is “one of the most important, most expansive, and most powerful tools in the 90s, in almost any business area...” [THO94]. Initially telecommunication networks provided only the most basic functionality in terms of setting up and disconnecting telephone calls. Next the Intelligent Network concept was introduced and brought services oriented to facilitate the placing of phone calls. Examples are Abbreviated Dialing, Call Forward, Call Waiting. Intelligent Networks are the predominant architecture in the today Telecommunications. They provide for rapid and flexible implementation, deployment and management of new telephony services. Intelligent Networks are based on the Circuit Switched paradigm, which relies on establishing a dedicated end-to-end connection between the communicating parties. Around 1970s another technology, the Packet Switched Technology, started to emerge and became the basis of the Internet - the most popular packet switched network today. Carrying voice over Internet is one of the most promising technological trends in modern Telecommunications in terms of reduced deployment and maintenance costs, efficiency of bandwidth use, new services, seamless integration of voice and data communications, etc. Internet Telephony is a reality today, but it is in its infancy. We witness an explosion of research efforts in this area.

One of the issues, which is the most important for the success of this new telecommunication technology is the adoption of an Internet Telephony standard supported by the major computer industries. Currently there are two major competing standards: the ITU-T H.323 protocol stack and the IETF Session Initiation Protocol.

H.323 is an umbrella recommendation that sets standards for multimedia communications over packet switched networks. It takes the traditional circuit switched approach to telephony signaling based on the ISDN signaling protocol and the earlier H-series recommendations.

SIP on the other hand takes a more lightweight and inherent to Internet Telephony Internet approach based on HTTP.

Which one will become the Internet Telephony standard? Whether it will be the first to come and in some extent commercially deployed, but complex H.323 or the newly proposed lightweight SIP? This question will be answered by the future, but for this much more research and implementations are needed.

Another issue that is substantial not only for Internet Telephony, but for Telecommunications in general, is the set of Value Added Services it is able to offer to its subscribers. We mentioned that Internet Telephony enables the creation of new class of services. They combine the best characteristics of voice communications and data processing. Examples are Web-enabled call centers, collaborative whiteboarding, remote teleworking [ITRT98]. But what about the variety of Value Added Services that Intelligent Networks currently offer to the telephone user and to which he is used to? Whether and how Internet Telephony will provide for, let say, Abbreviated Dialing, which the subscriber has been using with the circuit-switched telephony for quite some time? The service provisioning is an open issue in Internet Telephony and is in the focus of a lot of research and implementation efforts. It also gives the motivation for the study presented in this thesis.

Value Added Services in Internet Telephony is very wide topic with many directions to be explored. Here we will take the path of SIP based Internet Telephony and we will present an analysis of the common approaches to Value Added Services in SIP environment. Next we will study the provision of Value Added Services in SIP Internet Telephony from a different perspective, notably from the perspective of reusing the already existing IN VAS implementations. For this purpose we will suggest a hybrid SIP-IN network scenario, and will identify the design issues that need to be solved. Focusing on one of them – the communication interface between the interacting SIP network and IN entities, we will present an architecture design of a lightweight protocol built as a SIP extension.

The thesis is organized as follows:

Chapter 1 introduces the basic concepts of Intelligent Networks and presents the semantics of the IN/WIN Value Added Services. Chapter 2 gives an overview of the Internet Telephony and outlines the two competing standards – H.323 and SIP. Chapter 3 focuses on SIP and describes SIP functionality, operation, architecture entities and messages. Chapter 4 studies the aspects of Value Added Service implementation in a pure SIP environment. Chapter 5 presents a hybrid SIP-IN approach to the service implementation and identifies the open issues related to it. One of them – the communication interface between the SIP and IN entity, is pointed as a key issue and is further detailed. Chapter 6 is the core of this work and presents an architectural design of a lightweight protocol supporting the SIP-IN nodes communication. The protocol, built as an extension to SIP, is designed in terms of architectural entities, extension headers, and protocol operation. Chapter 7 outlines a possible functional model of the architecture entities implementation and some relating issues. The thesis closes with a conclusion and some hints for future work.

Chapter 1

Advanced services in classical telephony

When we turn back to the early days of the telephony, we can see the long way that telecommunication technology has come in around 100 years.

The telephony sophistication, to which we are all used today, is result from the fact that both the telephony subscribers and providers strive for more new advanced services. These services add value for the subscribers by making it easier to use the phone and to complete a call and generate revenues for the network providers. [THO94]

The first chapter briefly follows the evolution of telecommunication networks. It introduces the Intelligent network (IN), which is the predominant architecture for the telephony of today and defines the services commonly offered in IN. The package of services offered by the Ericsson Cellular Mobile platform CMS 8800 is described as the basis for the further study.

1.1 Evolution of the classical telecommunication networks

The milestones that mark the evolution of telecommunication networks are:

- 1878

Plain Old Telephone Service (POTS) is born with the first large city exchange in San Francisco [FAR].

The POTS environment represented a network of hardware switching systems, which provided only the basic telephony services – call establishment and call termination.

- 1965

Network deployment of Stored Program Control (SPC) switching systems begins.

They allow for additional services (also called “supplementary services” or “advanced services” in order to distinguish them from the basic telephony services) to be provided by the switching systems. Example of such services are call forwarding, toll free calling, and call distribution. SCP was a major step forward because now the service logic was programmable, while in the past it was hard-wired. Nevertheless, this service logic concept is not modular. Each supplementary service represents monolithic, non-reusable software that requires a modification in the basic call process of the switch. The “intelligence” and the switching functions are integrated in the telephone exchange. This means that the introduction and maintenance of services is a complex and expensive task.

- 1980s

Intelligent Network (IN) concept is introduced.

The Intelligent network approach consists of centralizing the service data and service program (figuratively referred to as “intelligence”) outside the switches in “intelligent nodes” – computer nodes distributed throughout the network. The “intelligent” node is able remotely to control the establishment of call connections at the request of switches. [MAG96].

This concept provides the network operator with more flexibility to develop and control services efficiently so that new capabilities can be rapidly introduced into the network.

The Intelligent Network is the new technology that dominated the trends in the Telecommunication industry for the last two decades. It is bound to be the starting point for almost all research done in this area. The basics of the IN architecture are presented in the next subsection.

1.2 IN platform

The Intelligent Network platform constitutes of a technical platform, which contains components as local and transit nodes, transmission systems, service control points (SCP), service switching points (SSP) and so on. The IN platform also comprises the administrative and management platforms that are required to fulfill the rapid implementation of new services [THO94]. The IN platform is presented in more detail below.

1.2.1 Basic elements of the Intelligent network platform

The basic elements of an IN platform and their interaction during a Freephone service execution are illustrated on Fig. 1.1.

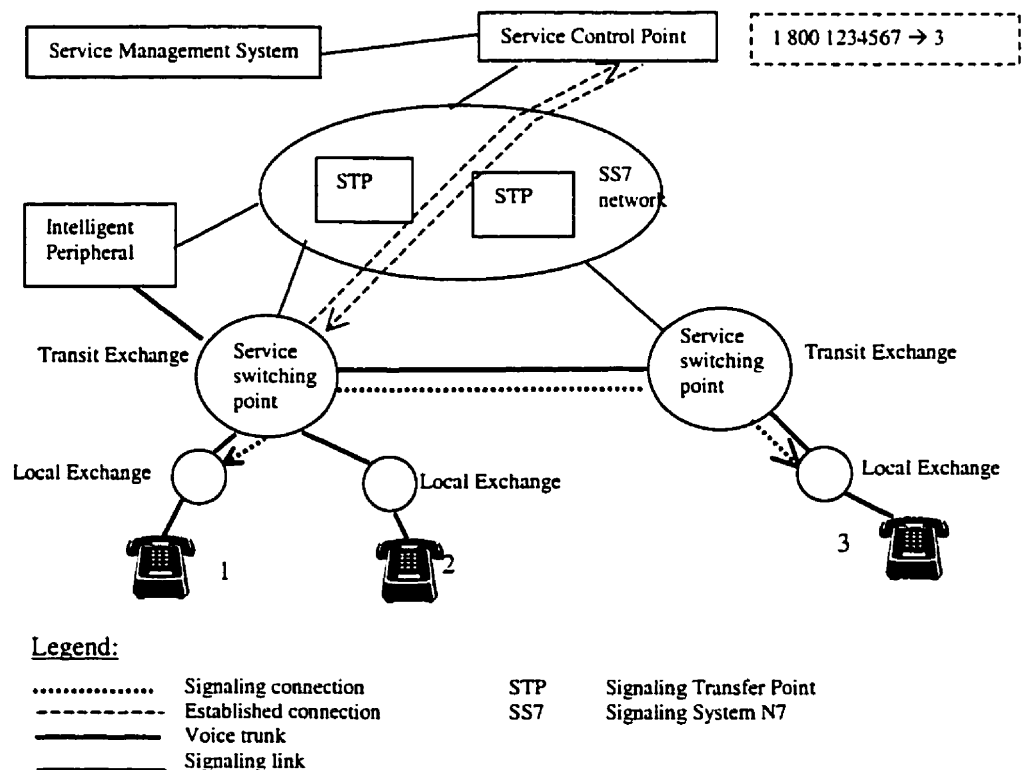


Fig.1.1

IN-based Freephone service implementation (adapted from [MAG96])

As seen in Fig.1.1 the IN platform comprises the following basic components: Service Switching Point (SSP), Service Control Point (SCP), Intelligent Peripheral (IP), Signaling System N7 (SS7), and Service Management System (SMS). The functionality of each of them is briefly described below:

SSP – Service Switching Point

The SSP is an IN enabled telephone switch. It provides the end users with access to the network and performs any necessary switching. In addition, SSP has the ability to detect requests for IN-based services and establish communications with the IN service logic located at the SCPs. [INT99]

SCP - Service Control Point

SCP provides the service control. It represents a computer system running a database of service logic and service data, able to provide call-handling information.

There are two basic parts a SCP - generic platform developed by SCP vendors and service logic application programs. The SCP is connected to SSPs by the SS7 network.

IP - Intelligent Peripheral

Under the control of SSP and SCP, the IP provides resources such as customized and concatenated voice announcements, and voice recognition. The IP contains a switching matrix to connect users to these resources.

SS7-Signaling System Number 7

SS7 is the protocol that runs over the Common Channel Signaling Network (CCSN). The SS7 network consists of packet data links and packet data switching systems called signaling transfer points (STPs).

The SS7 network separates the call setup information and talk path from the common trunks that run between switching systems. The call setup information travels outside the common trunk path over the SS7 network. The dashed lines on Fig. 1.1 represent an example of Freephone setup with SS7.

SMS – Service Management System

SMS is used for operation and maintenance tasks, such as provision of services, collection of statistics, supervision, maintenance of SCP, and software downloading.

Fig. 1.1 gives an idea of a typical Freephone service call setup. The Freephone service steps, illustrated on the figure are:

1. The Local Exchange node recognizes a free call service.
2. The Local exchange routes the call to an SSP, which is often a Transit Exchange.
3. At the SSP the call is suspended, while a setup to the SCP is made via SS7 network.
4. At SCP the Freephone number is translated to the destination number for the Freephone call.
5. When SSP receives the destination number back, the normal call is resumed to connect to its local exchange node.

The Freephone service example illustrates the following characteristics of the IN:

- The control of the service is centralized at a single point in the network – the SCP. SCP has a total overview of the execution of the service – including logic, data, and management.
- If the service is changed, mainly SCP is affected.
- It becomes possible to put the “intelligence” in more than one place in the network.
- Provided there is a good and secure network signaling, we can put the control facility of the service anywhere in the network.

These characteristics of the Intelligent Network formulate its main advantages over the old Stored Program Control technology.

1.2.2 Advantages of IN

What does the Intelligent Network concept offers more than the Stored Program Control technique?

- It provides an open platform for uniform creation, introduction, control and management of advanced services.
- It allows a wide variety of services to be provided to customers independent of the underlying network technologies.
- It reduces the time required for service creation, deployment, and customization.

Today INs [MAG96] are deployed all around the world and many IN services are and more will be offered. That makes the IN advanced services a necessary starting point for exploring the Telecommunication networks and services for the future.

1.3 Semantics of the Advanced Telecommunication Services commonly offered in IN

The following overview of services, is based on Capability Set 1 (CS1) and Capability Set 2 (CS2). The term Capability Set refers to the set of services and service features that can be commonly constructed in particular evolution phase of IN.

The CS1, CS2 are a good source for getting a comprehensive view of the advanced services, which could be offered by IN.

For more a systematic representation, the services can be classified in few categories. (Note that one IN service may fall into several categories). The classification presented below follows the one given in [MAG96]. Since different classification criteria can be used, this service classification is not unique.

1.3.1 Number Translation Services

In this category are included services, that involve destination number resolving or management of control information needed. These services may be differentiated in a few subcategories: Forwarding and User Location services, Personal Mobility services, End User Control services, Conferencing, and Other Call Translating services.

Forwarding and User Location services

In this subcategory fall the following services: Call Distribution, Call Rerouting Distribution, Call Forwarding, Mass Calling, Universal Access Number, and Universal Personal Telecommunications.

Call Distribution: Routes the incoming calls to different destinations according to specified criteria.

Call Rerouting Distribution: If a trigger condition is encountered, reroutes the incoming calls (busy, queue overload).

Call Forwarding: Forwards calls to another number. It may be:

- non-conditional
- conditional (busy or don't answer) - also called Selective Call Forwarding

Mass Calling: Provides capabilities for instantaneous, high volume traffic routing to one or more destinations depending on specific conditions.

Universal Access Number: Allows a subscriber with several network addresses to be reached with a single unique address.

Universal Personal Telecommunications: Allows the user to access telecommunication services via a unique personal telecommunication number (PTN) across multiple networks and at any network access.

Personal Mobility Services

The only service in this category is Universal Personal Telecommunications.

Universal Personal Telecommunications (UPT): Provides the subscriber with mobility. Based on a unique personal telecommunication number (PTN).

End User Control Services

Services which involve end-user control are Follow Me Diversion, and User Defined Routing.

Follow Me Diversion: Allows the user to control remotely the redirection of calls.

User Defined Routing: Allows the subscriber to determine the routing for outgoing calls according a routing preference list.

Conferencing

A service from this subcategory is Conference Calling.

Conference Calling: Enables a group of users to be connected into a multi-party call.

Other Call Translating Services

This subcategory comprises translating services, which do not fall in any of the previous subcategories. It includes Abbreviated Dialing, Freephone, and User defined routing.

Abbreviated Dialing: Allows abbreviated versions of telephone numbers.

Freephone: Allows reverse charging for a unique number (seen also as Billing service).

User defined routing: Allows personal routing schemes for outgoing calls (also End User Control Service).

1.3.2 Screening Services

The services involve flexible screening and possible restriction of the call establishment. They may incorporate some translation capabilities, but the service focus is on the screening capabilities.

Originating Call Screening: The subscriber is able to screen outgoing calls according to a list. The user may override the restriction with a PIN.

Terminating Call Screening: The subscriber can construct a list to specify whether the incoming calls are allowed or not.

Security Screening: Doesn't allow non-authorized access to the subscriber network.

1.3.3 Billing Services

These services take advantage of the flexible charging capabilities. They may incorporate some number translation services, but the service focus is on special charging or billing. [MAG96]. The services presented in this category are: Account Card Dialing, Automatic Alternative Billing, Credit card calling, Freephone, Premium rate, and Split charging.

Account Card Dialing: The subscriber can use a telephone card to make phone calls from any card-reading telephone.

Automatic Alternative Billing: Enables charging a user for a call by the user's account from any telephone, i.e. the call charge does not refer either to the calling line or to the called line. An account code and PIN are allocated to the service user. These services allow the user (the calling party) to ask another user (the called party) to receive the call at her or his expense.

Credit card calling (CCC): The user may call from any normal access interface to any destination and is charged to an account specified by the CCC number.

Freephone (use of Reverse Charging): The subscriber is able to reverse charging by accepting charges for incoming calls. (also Translating service).

Premium rate: The service provides the subscriber with premium rate number. A user who calls this number is charged at a special rate for both the call and the service/information obtained by the call. The network operator collects the revenue and shares it with the service provider.

Split charging: The calling as well the called party is charged for part of the call.

1.3.4 Other Services

These are service, which can not be included in any of the other categories. They may incorporate flexible routing, charging, screening, number translation, but the service focus is on something particular.

Call completion to busy subscriber (CCBS): Allows a calling user encountering a busy destination to be informed when the destination becomes free.

Call Waiting: The called party is notified that another party is trying to reach her while she is busy.

Customized ringing: Distinctive ringing based on caller identification.

Malicious Call Identification: Allows controlling the logging (record) of incoming calls (seen also as a screening service).

Virtual Private Network: Provides private network capabilities by using public network resources. The subscriber lines, connected to different network switches, constitute a VPN including capabilities such as private numbering plan and Call Transfer.

Televoting: Enables voting via the network.

Call Transfer: Allows the user to place a call party on hold and to enter a new destination number.

Call Hold: Allows user to place a call on hold and to play an announcement or to initiate a new call. After that the user can resume the first call.

Multimedia: Allows receiving of integrated data, voice, image, and video [ITAP98].

Message store and forward: Enables a user to distribute a message to one or several destination users.

Hot Line: Allows a user to place a call without providing the called party information required by the network. This routing information is stored in the network by subscription.

The services described above represent part of the set of services that can be constructed in an Intelligent Network if that network conforms to the accepted IN standards. As far as

they represent the services that potentially can be implemented in an IN, they give a theoretical overview of the IN based advanced services.

Intelligent Networks are now an expanding area within the telecommunication industry . There are more and more competitors on the market each providing different solutions and set of services. The Table 1.1 presents a few of IN/WIN (Wireless IN) solutions and some of the services they offer as a prepackaged set or services that can be potentially built.

	Ericsson (WIN) CMS 88000 product family	DCS (IN/WIN) INfusion product family (prepack. Portfolio)	Torch Telecom (IN)	Lucent Technologies Service Node (WIN)	NOKIA Mobile VAS
Short Message Service		X		X	X
Call Screening	X	X			
Debit Card		X			
Call waiting			X		
Group pick up			X		
Group 'hunting'	X		X		
Three-way calls			X		
Caller line identification			X		
Abbreviated dialing			X		
Freephone	X		X		
Automatic call distribution			X		
VPN	X			X	
Voice Activated Dialing				X	
Voice/Fax Mailbox				X	X
Paging Message Delivery				X	
Over-the-Air Activation Provis.				X	
Authentication Center				X	
Customer Control Access	X				
Bulk Number Translation	X				
Outgoing Call Allowance/Restr.	X				
Selective/Flexible Call Forwarding	X				

Table 1.1
Commercial IN/WIN service packages

The package of services, used as base for our further study, is the off the shelf package of services offered by Ericsson's Cellular Mobile System platform CMS 8800.

1.4 Semantics of the Telecommunication Services offered by CMS 8800

Before going into the semantics of the CMS 8800 Wireless Intelligent Network services, the Ericsson's Cellular Mobile Platform CMS 8800 will be briefly introduced.

In the world of mobile communication there are two major standards for wireless systems: AMPS and D-AMPS (TDMA).

AMPS stands for Advanced Mobile Phone Service and was one of the first ever analog wireless systems. It is still one of the most widely used technologies today (USA analog cellular standard).

TDMA stands for Time Division Multiple Access and refers to the specification IS-136 for advanced digital wireless services. It is also one of the world's most widely deployed digital wireless systems. D-AMPS (Digital Advanced Mobile Phone Service) is the USA standard.

CMS 8800 is the Ericsson's networks platform able to handle TDMA/AMPS standards for mobile systems. It covers:

- Hardware platform;
- Network administration & operation;
- Support for services.

CMS 8800 is being continuously updated to include evolving services and applications. For example, pre-paid calling, voice-activated services and CDPD (Cellular Digital Packet Data) can all be supported by the CMS 8800 platform.

The set of services described below is part of the Wireless Intelligent Network implementation for the CMS 8800 family. They may be classified in two main categories: Subscriber services and Network services.

1.4.1 Subscriber Services

Subscriber services are services, which can be offered to an individual user.

Depending on the point in the network they were triggered, the subscriber services fall in three subcategories: A-number services, B-number services, C-services.

A-number IN Services

The A-number IN services are also called originating services, because they are triggered by the calling party (the A number). In this group fall the following services: Customer Control Access, Bulk Number Translation, Private Numbering Plan, Outgoing Call Allowance, Outgoing Call Restriction. Their semantics is shortly presented below.

Customer Control Access (CCA): Allows the user to modify certain service subscription data using their cell phones. CCA is available for Outgoing Call Allowance, Outgoing Call Restriction and Selective Call Forwarding.

It is End User Control Services (as defined in IN).

Bulk Number Translation (BNT): Allows defining abbreviated dialing code for a range of numbers. The subscriber dials the code and a suffix to compose the destination number.

It is Call Translating Service (IN).

Private Numbering Plan(PNP): Allows subscribers to use abbreviated codes to reach frequently called numbers.

It is a translation service analogous to Abbreviated Dialing (IN).

Outgoing Call Allowance (OCA): Calls placed from the subscriber phone are routed to the dialed number only if it is on the screening list.

It is Screening Service, analogous to Originating Call Screening (IN).

Outgoing Call Restriction (OCR): The placed calls are verified against of screening list of unauthorized numbers.

It is Screening Service, analogous to Originating Call Screening (IN).

B-number IN Services

The B-number IN services are also called terminating services, because they are triggered by the reception of the destination number (the B-number). This group includes the Selective Call Acceptance, Selective Call Rejection, and Selective Call Forwarding services, and are introduced below.

Selective Call Acceptance: Calls are routed to the subscriber only if the calling party is in the list of authorized numbers.

It is Screening Service, analogous to Terminating Call Screening (IN).

Selective Call Rejection: Calls are routed to the subscriber only if the calling party is not in the list of unauthorized numbers.

It is Screening Service, analogous to Terminating Call Screening (IN).

Selective Call Forwarding: Allows the subscriber to designate where the calls from the numbers on the screening list and not on the screening list to be forwarded.

It is a Screening and Translation Service (IN).

CIN Services

The CIN services are called transferring services, because they are triggered by a call transfer conditions in the HLR/SCP, such as call busy or no-answer. Only the service Flexible Call Forwarding is presented in this group.

Flexible Call Forwarding: Allows the subscriber to forward their calls to various destinations based on subscriber-defined forwarding schedule.

It is a translating service and is similar to Destination Call Routing.

1.4.2 Network Services

The Network Services apply to the network and cannot be provided for an individual user. The Network services are: Toll Free Calling, and Wireless Virtual Private Network.

Toll Free Calling: The owner of Toll Free Number offers mobile users the capability to contact them free of charge.

It is a translating Service and is analogous to Freephone (IN).

Wireless Virtual Private Network: A collection of services that allows operators to provide subscribers with private network capabilities such as Private Numbering Plan, Bulk Number Translation. Some of the other services that it can optionally include are Outgoing Call Allowance, Outgoing Call Restriction and Selective Call Forwarding.

WVPN is created by assigning users to a Selective User Group, thus giving them access to the services provided by WVPN.

1.5 Conclusion

As seen from the international standardization documents (CS1, CS2), the telecommunication industry today is able to offer an overwhelming variety of services to the end users.

The CMS 8800 WIN services, as a subset of them and as a commercially offered package, are our motivation and reference point for pragmatic exploration of the new trends in providing value added telecommunication services. One of the most promising avenues for further development is the Internet Telephony.

Chapter 2

Internet Telephony

Internet Telephony refers to communications - voice, facsimile, and/or voice-messaging applications - that are transported via the Internet rather than the public switched telephone network (PSTN) [ITAP98].

This chapter gives high level overview of the Internet Telephony and its strengths. It shortly presents the protocol stack on which the Internet Telephony is built. Next it focuses on the existing Signaling Protocols - protocols which provide for the deployment of advanced Internet Telephony services. The H.323 and SIP protocols are outlined.

2.1 Overview

The PSTN provides users with dedicated, end-to-end circuit connections for the duration of each call. The PSTN also provides access to intelligent network services using the SS7 protocol [IP/IN99].

Unlike the circuit-switched PSTN, packet-switched IP networks provide shared virtual circuit connections between users. IP packets are routed to the destination IP address contained within the header of each packet. Packets may travel over separate network paths before arriving at their final destination for reassembly and resequencing. Transmission speed between any two users can change dramatically based on the dynamic number of users sharing the common transmission medium, their bandwidth requirements, the capacity of the transmission medium, and the efficiency of the network routing and design [IP/IN99].

The Internet Telephony has evolved rapidly in the past few years. It offers a range of advantages over the traditional PSTN network:

- The Internet Telephony has the ability to combine and transfer multiple media over the same network infrastructure.
- It does not depend on the underlying physical connection as far as it runs IP.
- The packet-based IP telephony is more efficient than the circuit-switched telephony from a network utilization standpoint. In the circuit-switched networks a full end-to-end circuit is allocated for the duration of the call. IP telephony also saves network resources through compression, reducing 64 kbit/s PCM streams down to typically 5-8 kbit/s for voice or 14.4 kbit/s for fax. Further savings are made through silence suppression.
- The Internet Telephony end systems are much more intelligent in PSTN. This allows the implementation of services to be moved toward the edge, without requiring explicit support within the network.
- The computer telephony integration is inherent to the Internet Telephony. It allows calls to be placed from PC to PC, from PC to phone, from phone to PC and phone to phone (Fig. 2.1).

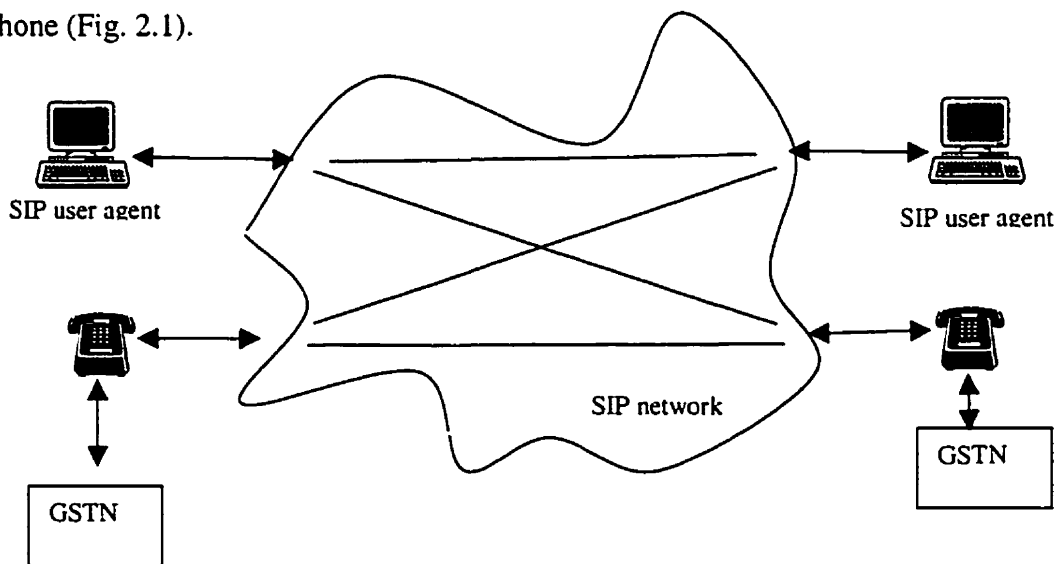


Fig. 2.1

Call connections in Internet Telephony

But what could make the Internet Telephony a strong competitor for a place in the world Telecommunications is its potential for faster and easier implementation, deployment and support of even more services than the classical IN telephony offers today [LSL99].

Unlike circuit-switched telephony, Internet Telephony is built on a range of packet switched protocols, presented in the next subsection.

2.2 Protocols specific to Internet Telephony

To be able to explore the Internet Telephony we need to have a good understanding of the protocol architecture on which it is based (Fig.2.2 [SIT98]).

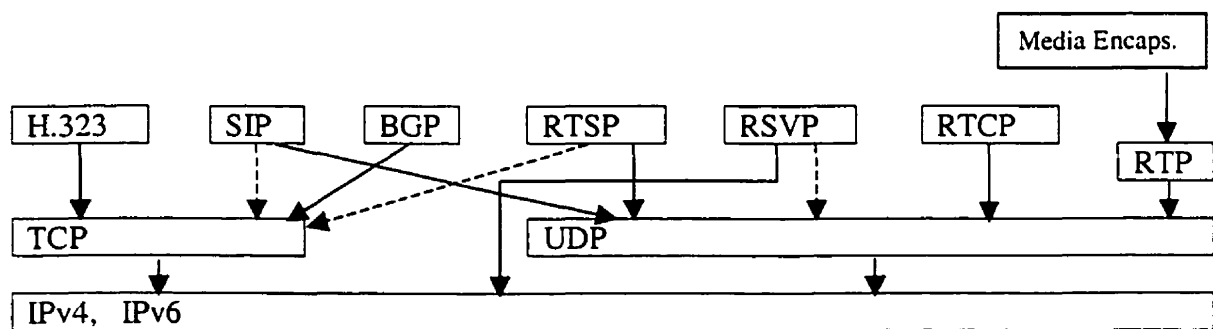


Fig. 2.2
Internet Telephony protocol stack

The protocols from the Fig 2.2 have the following functionality:

RTP (Real-Time Transport Protocol) – to carry voice and video data

RTCP (Real Time Streaming Protocol) – for control of streaming media

BGP – for routing (not specific for Internet Telephony)

RSVP or other reservation protocols – for resource reservation

SIP (Session Initiation Protocol) – for signaling and control

H.323 – for signaling

The next subsection presents the protocols, which are specific to the Internet Telephony – SIP and H.323.

2.3 Signaling protocols

The need for signaling and control distinguishes the Internet Telephony from the other Internet applications. A message can be delivered to the mail server and retrieved and answered by the recipient at his convenience. A telephone call requires real time signaling for the parties to find each other and to signal their availability or desire to communicate.

This is the reason the signaling protocols are of particular interest when considering Internet Telephony. These protocols are the basis for providing advanced services, such as mobility, call distribution/forwarding, universal numbers, multi-party conferences, voice mail, etc. [SATS98].

Two signaling protocol families have emerged: the ITU-T H.323 and the IETF Session Initiation Protocol (SIP).

2.3.1 H.323 protocol

H.323 is an umbrella recommendation from the International Telecommunication Union (ITU) that sets standards for multimedia communications over non-guaranteed bandwidth packet switched networks. The Internet and LANs using TCP/IP and SPX/IPX protocols running over Ethernet or Token Ring are examples of packet switched networks with non-guaranteed bandwidth [DCWP98], [BRA98].

H.323 is based on the ITU multimedia protocols, which preceded it – H.320 for ISDN, H.321 for B-ISDN, H.324 for GSTN terminals. The encoding mechanism, protocol fields and basic operation are simplified versions of the Q.931 ISDN signaling protocol [CSH98].

H.323 provides various levels of multimedia communications:

- voice only;
- voice and video;
- voice and data;
- voice, video, and data.

2.3.2 Session Initiation Protocol

SIP is a protocol for signaling and control for Internet Telephony. It provides for session creation, modification and termination, capability exchange, and conference control. [CSH98], [rfc2543]. It is currently under development in the MMUSIC working group of IETF.

SIP is client server protocol and it is based on exchanging messages between them.

There are two types of messages – request and response. The messages are text based and use the ISO 10646 character set. It reuses the message syntax and some of the header fields from HTTP 1.1 [rfc2543].

Before presenting SIP in detail let us first mark some of the basic reference points for a comparative study of the two signaling protocols.

	H.323	SIP
Content	Protocol suite	Single protocol
Representation	Binary	Text based
Designed for:	Signaling protocol for multimedia communications in general.	Signaling protocol for Internet only.
Media supported	Voice, video, data	Voice, video, data
Transport protocols	On top of TCP	On top of TCP or UDP
Codecs allowed	Imposes restriction on the video and audio codecs used in terminals.	Can work with any codec. The end systems can negotiate about the media exchanged and the encodings supported.
Services implementation	Separate standard is needed for any additional service.	No need for service standardization.
Conference control	Centralized conference control	Distributed conference control
Implementations Complexity	Complex	Simple
Market tested	H.323 platforms exist on the market.	No existing SIP deployments yet.

Table 2.1
H.323/SIP comparison

The Table 2.1 gives very high level comparative overview of SIP and H.323. The comparison between them is an open topic in the Internet community. Both protocols are currently being developed and new features are being added to both SIP and H.323.

2.4 Conclusion

H.323 is the protocol that came first to the market. Compared to it SIP is “young technology” [BER98]. But it is much more simple and “uses the Internet in all its richness of features as it is” [Ipnet, 98].

The question, which standard will dominate the Internet Telephony in the future or how the existing standards will coexist, is in the center of ongoing debates. The thing, on which everybody agrees, is that much more research and implementations are needed. Following this line, the next chapter is devoted to study of the Session Initiation Protocol and the support that it provides for Value Added Services.

Chapter 3

Session Initiation Protocol

This chapter gives an overview of a SIP architecture. It starts with a description of its building components – end systems and network servers and the tools that SIP offers for call signaling and control. Next it presents the basic operation of a SIP system.

3.1 Architectural Components

Before going into more detailed study of SIP and its operation we need to have a good understanding of the basic architectural components which are commonly present in a SIP based system (Fig. 3.1) – user agent client/server, proxy, redirect, registrar server.

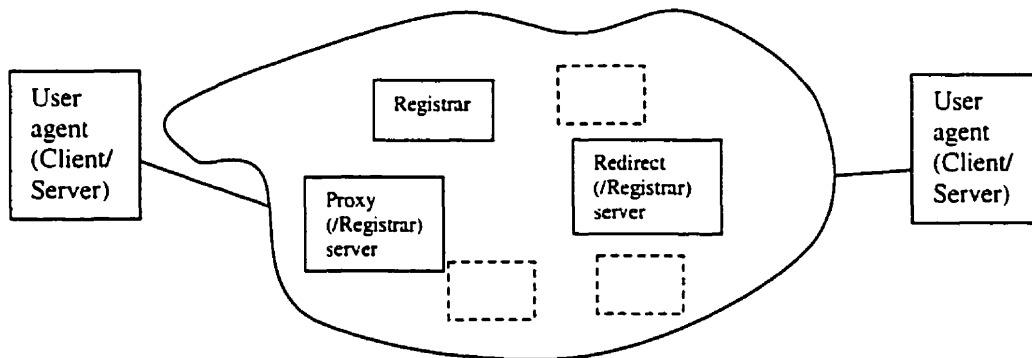


Fig. 3.1

Architecture components of a SIP system

The basic SIP architectural components have the following functionality:

- User Agent Client/Server (UAC/UAS)

The User Agent Client/Server is a SIP-enabled end system of a call participant. It resides on the host where the user is situated. It is able of querying the user about what to do with

the call – accept, reject, forward [SIT98]. As a call participant may either generate or receive requests, the User Agent includes a protocol client and server [ITAP98].

- Proxy Server

The proxy server receives a request and forwards it towards the next hop server, which may be another proxy server, redirect server or the user agent/client server. [ITAP98]. It acts also as a client, issuing one or more outgoing requests triggered by each incoming request [RSUA98]. A proxy server can be either stateful or stateless. When stateful the proxy remembers the incoming request, which generated the outgoing request, and the outgoing request. A stateless proxy forgets all the information once an outgoing request is generated [rfc2543].

- Redirect Server

It informs the client of the address of the next hop server, so that the client can contact it directly [ITAP98]. The redirect server does not maintain any call state [RSUA98].

There is no protocol distinction between a proxy server, redirect server or user agent. The distinction lies only in the function: a proxy or redirect server cannot accept or reject a request, while a user agent can. This is similar to HTTP model of clients, origin and proxy servers [ITAP98].

Servers should be able to operate in either mode, preferably configurable on a per user or domain basis [RSUA98].

- Registrar Server

The Registrar server allows the users to register their current location. It may also offer other location services. More often it is co-located with a redirect/proxy server, but it may be a separate network entity.

When a call from a user agent server has to be terminated to another network (e.g. SIP client wants to connect to PSTN phone number) then additional architecture components participate in the call process. These are Wide Area Service Location Server (WASRV), Internet Telephony Gateway (ITG).

- Wide Area Service Location Server (WASRV)

The SIP client sends a request to the WASRV with the telephone number to contact, user preferences about billing, etc. The WASRV queries its database, and returns the address of an appropriate gateway [ITAP98].

- Internet Telephony Gateway (ITG)

ITG is needed to terminate Internet calls to GSTN [ITAP98].

A SIP system that uses the services of a WASRV server and ITG is presented on Fig. 3.2.

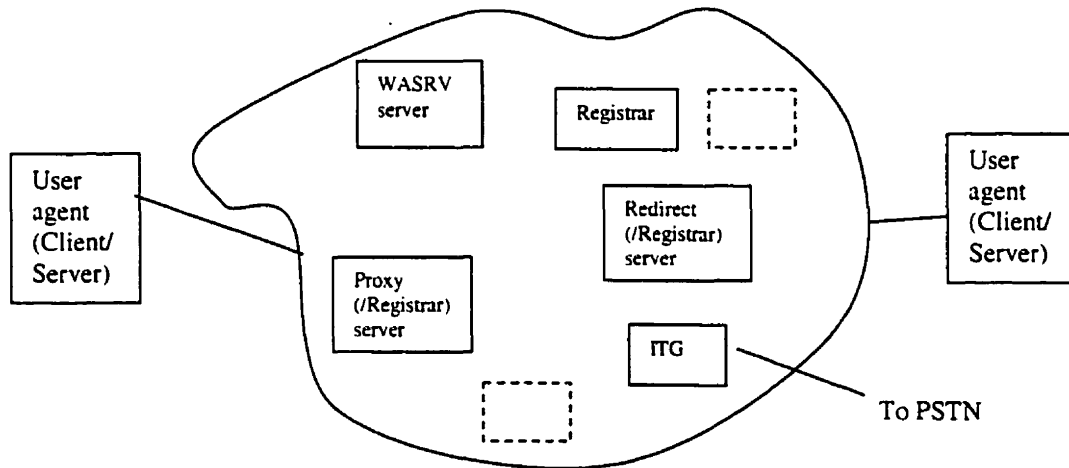


Fig.3.2

Architecture components of a SIP system connected to other networks

After outlining the basic architectural components, let us have a closer look at the transport protocols that can be used for communication between them.

3.2 Underlying transport protocols

SIP may use any underlying transport protocol. It can directly use any datagram or stream protocol, with the only restriction that a whole SIP response or request has to be delivered

either in full or not at all. SIP can thus be used with UDP or TCP in the Internet and with X.25, AAL5/ATM, IPX, PPP elsewhere [ITAP98].

- UDP

In the case of UDP no connection state is required. This means that large, backbone servers can be based on UDP and operate in a stateless fashion, thus greatly reducing the memory requirements and improving scalability.

As SIP can use UDP, it supports native multicasting, allowing a single protocol to scale from sessions with two to sessions with million members [CSH98] and to multicast SIP requests for some signalling operations such as searching [SIT98].

- TCP

TCP significantly increases call setup delay – the three-way handshake adds 1.5 round trip time of delay. Also current TCP implementations are very conservative in their retransmission of initial SYN packet, with retransmit delay of 6 to 24 s. Thus, even a single packet loss will lead to unacceptable call setup delay.

TCP features such as flow and congestion control are not particularly helpful in the transport of SIP messages since they are likely to be short and sporadic. However, for firewalls and transport layer security protocols, the use of TCP may be required.

3.3 Addressing and naming

For user addressing, SIP uses an email-like identifier of the form “*user@domain*”, “*user@host*”, “*user@IP_address*”, or *phone-number@gateway*. Addresses of the form *phone-number@gateway* designate PSTN phone numbers reachable via the named gateway [SIT98].

3.4 SIP messages

As mentioned earlier SIP is based on exchanging messages between the SIP servers and clients. The SIP messages contain *Start line*, *Headers* and *Message body*. The *Start line* formulates the SIP message either as a request or as response. Following the graphic representation of the SIP message building elements given on the Fig. 3.3, the next subsections briefly describe the request/response messages, headers and message body defined in SIP.

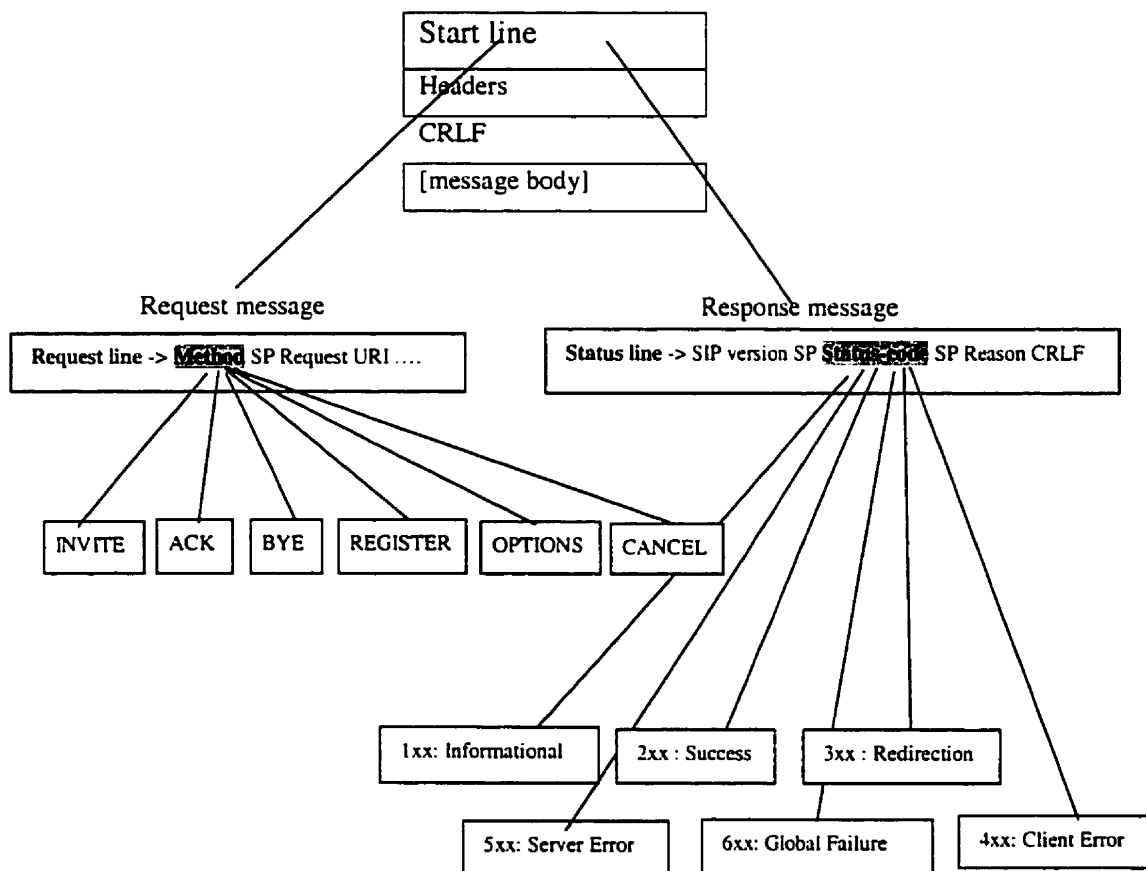


Fig. 3.3

SIP message elements

3.4.1 Requests

The requests may implement one of the six methods INVITE, ACK, OPTIONS, BYE, CANCEL and REGISTER (see Fig. 3.3), briefly presented below.

INVITE

It is used to invite a user to participate in a session. The message body contains a description of the session to which the callee is invited (description may include port for the media stream, type of codecs supported, etc). The parameters of an existing session may be changed by sending new INVITE, with the new session description.

ACK

It confirms that the client has received final response to an INVITE request. The ACK request may contain a message body with the final description to be used by the callee. Otherwise the description from the INVITE is used.

OPTIONS

It is used to query the server about its capabilities. The server may return a status reflecting how the user would respond to a call (e.g. "busy").

BYE

It indicates a wish for terminating the call. May be issued by both parties.

CANCEL

This request cancels a pending request. It may be used to terminate parallel searches when the user is reached on one of the locations.

REGISTER

The client issues this request when the user wants to register his address with a new location. There is first party registration (the entity that issues the request is the same as the one being registered) and third party registration (the entity issuing the request is different from the entity being registered).

3.4.2 Responses

In the responses the most informative element is the status code. The status code is a 3-digit integer result code that indicates the outcome of a request. The first digit defines the class of the response. SIP responses are extensible, i.e. more codes can be added to each class. SIP applications are not required to recognize all of the codes, but they must recognize the class of the response.

The Table 3.1 gives an overview of the status code classes defined in SIP.

1xx:	Informational	Request received, continuing to process the request.
2xx:	Success	The action was successfully received, understood, and accepted.
3xx:	Redirection	Further action needs to be taken in order to complete the request.
4xx:	Client Error	The request contains bad syntax or cannot be fulfilled at this server.
5xx:	Server Error	The server failed to fulfill an apparently valid request.
6xx:	Global Failure	The request cannot be fulfilled at any server.

Table 3.1
Response classes

3.4.3 Headers

SIP header fields are similar to HTTP header fields in both syntax and semantics. The header fields may be required, optional or not applicable for the methods described above. Depending on where the header fields can be or are most commonly used, they can be classified in few categories.

- General Header Fields

They apply to both request and response messages. Some of the most important are:

Fom	Indicates the initiator of the request.
To	Indicates the recipient of the request.
Cseq	Sequence number, used to identify duplicate messages.
Call-ID	Uniquely identifies a call. It is created by the originator of the call and used by all call participants.
Via	Indicates the path taken by the request so far.

Table 3.2
Examples of general header fields

- Entity Header Fields

The entity header fields define meta-information about the message body. Some of the most important are Content-Length and Content-Type.

Content-Length	Indicates the size of the message-body in decimal number of octets.
Content-Type	Indicates the media type of the message-body.

Table 3.3
Examples of Entity header fields

- Request Header Fields

They allow the client to pass additional information about the request and about itself.

Accept	Indicates the media types acceptable in the response. It is used only in INVITE, OPTIONS and REGISTER requests.
Route	Determines the route taken by the request.
Priority	Indicates the urgency of the request.
Require	Indicates the options that the client expects the server to support in order to properly process the request.

Table 3.4
Examples of Request header fields

- **Response Header Fields**

They allow the server to pass additional information about the response, which can not be placed in the Status line.

Contact	Provides a URL where the user can be reached for further communication. It can appear in INVITE, ACK, and REGISTER requests or in 1xx, 2xx, 3xx, and 485 responses.
Retry-After	Indicates how long the called party or service is expected to be unavailable.
Allow	Informs the recipient of the valid methods associated with a resource.

Table 3.5
Examples of Response header fields

3.4.4 Message body

The message body is an optional part of the SIP messages.

The message body of the INVITE, ACK and OPTIONS requests (if present) always contains a session description. The use of the message body of the REGISTER requests is currently studied and is expected to contain SIP CGI or CPL (Call Processing Language) scripts. The BYE request must not contain message body.

A message body may also be present in all response messages. The response message body may contain request progress information (in 1xx responses), session description (in 2xx responses to INVITE request), alternative destinations or services (in 3xx responses) or human readable information about the reason of failure (in responses with status code 4xx or greater).

From this section we learned about the SIP messages and their building components. Next we will look at the basic operation of a SIP system.

3.5 Basic Operation

The most important SIP operation is that of inviting a new participant to a call [SIT98]. When inviting a new participant, a caller first has to locate the appropriate SIP server and then send to it an INVITE request. The process of SIP server location is described below.

- Locating a SIP server

When issuing a request, the client either sends it to a locally configured SIP proxy server or sends it to the recipient's server. For the latter case the client must determine first the IP address and port of the server to which to send the request [SIT98]. These two cases are explained in more detail below.

Case 1:

The address of the SIP proxy or redirect server to which the user agent client sends the INVITE request is locally configured in the user agent (Fig. 3.4).

If the SIP server contacted is a proxy, then the communication is only between the user agent and the proxy server (see Fig. 3.4, messages 2', 3', 4').

If the locally configured server is a redirect server, then the redirect server returns the address of the callee and the caller's user agent contacts it directly (Fig. 3.4, messages 2, 3, 4).

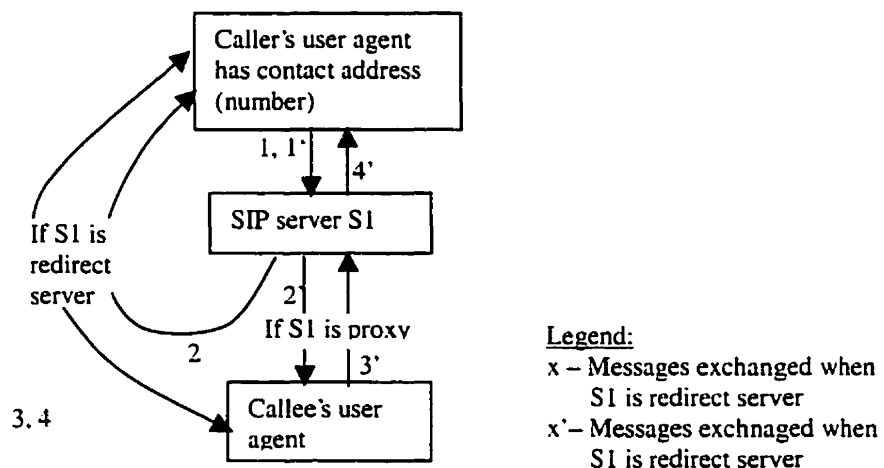


Fig. 3.4

Request sent to a locally configured SIP server

Case 2:

In this case the client must determine the protocol, port and IP address of the server to which the request must be sent. The process is described on the Fig. 3.5.

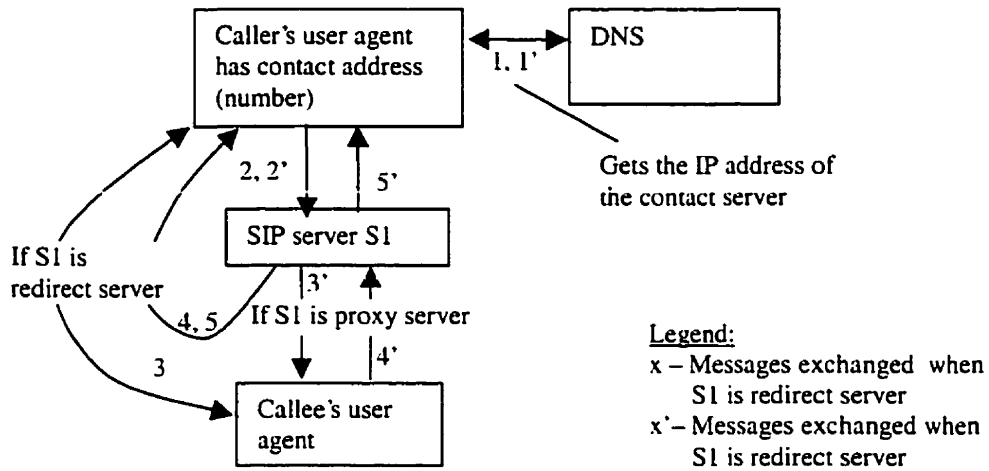


Fig. 3.5

Request sent to the IP address and port of the recipient's server

Besides the location of a SIP server to which the INVITE is to be sent, there is another aspect of the SIP operation relating to the support of user mobility – the user location. The operation of a SIP server for locating a user is summarized below.

- Locating a user

A user may move between a number of different end systems over time [SIT98]. His locations can be dynamically registered with a Registrar server. The user location may also be determined using a Location server.

The Registrar server is an optional SIP architectural component (described in section 3.1).

A Location server is a non-SIP entity. It may use one or more protocols (such as *finger*, *rwhois*, multicast protocols or other mechanisms) to determine where the user might be reachable. The location server may return several locations, because the user is logged on several hosts simultaneously [SIT98].



SIP server operation for user location

3.6 Conclusion

37

Chapter 4

Value Added Services implementation using SIP

SIP follows a generic approach to defining Value Added Services. Rather than explicitly describing the implementation of particular service, it only describes the behavior induced by messages and provides a number of elements to construct the services – requests, responses, headers, message body. Therefore there are many different ways of implementing the same service in SIP.

Conceptually the service implementation can be considered in two aspects: service construction resulting from the service execution and service creation.

4.1 Service construction

The term “service construction” refers to the sequence of requests and responses generated as result of the service logic execution. We can distinguish two general approaches to service construction in SIP:

- Using messages with general header fields (no service oriented)

For example, it is feasible to use this approach in the construction of the *Far End Mute* service.

Far End Mute: allows a participant to stop receiving media.

To implement *Far End Mute* service, a participant need only send an INVITE to another participant, indicating a null set of receive capabilities for any of the media.

This will cause the other participant to cease sending media.

This service does not require specific service oriented headers.

- Using messages and service-oriented headers like Contact, Replaces,

Camp-on is an example of a service that requires service-oriented headers.

Camp-on: Allows the caller encountering a busy signal to wait until the line becomes free without having to make a new call attempt.

The message constructed as result of the Camp-on service execution contains a *Call-Disposition* header as described below:

The caller sends INVITE
Call-Disposition: queue

The “queue” parameter indicates that the call has to be queued rather than rejected.

If the callee is busy, the server queues the call. It may issue several lxx (informational) responses to update the caller about the status of the queued call.

When the callee becomes available, it returns the appropriate final status.

Most of the services can be implemented using either of the two approaches. We can illustrate this with *Call Forward* service.

Call Forward

A. Constructed with service oriented field.

When the decision for forwarding is made by a redirect server, then the *Contact:* header field is used. The *Call Forward* service execution and message construction is described below:

1. The caller's user agent client sends an INVITE request.
2. The INVITE request reaches the redirect server where the call forward service logic resides.
3. The redirect server determines the forward address according the service logic and service data
4. The redirect server sent to the caller's user agent a 300-class response and the forward address specified in a Contact: header field.
5. The caller sends an INVITE request to the forward address.

B. Constructed without service oriented fields.

When the service logic resides on a proxy server, then this service can be constructed without service oriented headers.

The service execution and message construction follows the steps below:

- 1 The caller's user agent client sends an INVITE request.
- 2 The INVITE request reaches the proxy server where the call forward service logic resides.
- 3 The proxy server determines the forward address according the service logic and service data
- 4 The proxy server sends an INVITE request to the forward address.
- 5 The responses are proxied back to the caller's user agent.

4.2 Service logic creation

The service logic refers to the functions/processes used to provide a specific service.

The service logic is implemented in a software program (Service Logic Program) that uses the service data to generate messages, which correspond to the service semantics.

Some of the services (Service Logic Programs) may reside on the end systems. A broad set of services, however best reside in a network device (proxy, redirect, registrar server), either because they are independent of the end device or because they need to be operational even when an end device is unavailable. Examples of such services are those involving user location, call distribution, behavior on busy, etc. [CPLR98].

We will present two possible approaches to the service logic creation: standard compliant and proprietary.

4.2.1 Standard compliant service logic creation

Using this approach the service logic is implemented in the form of a script, which is compliant to the adopted standard. In this case, the service logic can be created either by the user or by the service provider. If it is created by the user and needs to reside in the network, the service logic script has to be uploaded to the network server. When a triggering call arrives, the server has to be able to execute the script, and depending on the script output, to perform the corresponding signaling actions. This approach requires a language for service logic creation, servers able to execute it, and optionally means for uploading to the network.

There are two solutions to the problem of Internet Telephony service logic creation proposed by the IETF: SIP-CGI and Call Processing Language (CPL).

Common Gateway Interface (CGI) is one of the mechanisms that provide for service creation in web environment. Since it is an interface and not a language, it can be extended and applied to SIP for developing SIP based service creation environments.

SIP-CGI model is built upon the basic HTTP-CGI model and is presented in [CGIS98].

The Call Processing Language (CPL) is a recently proposed script language [CPL99] and will be shortly reviewed below.

CPL provides a way for the end user to create services. The services in a CPL script are implemented by defining the conditions under which the servers take specific signaling actions [CPL99]. CPL is designed to be:

- independent of operating system or signaling protocol
- without variables, loops, functions
- not able to run external programs

Since for the time being CPL is the only standard proposed for SIP based service logic implementation, we will consider CPL only and we will refer to the standardized approach as the CPL approach.

While CPL the approach to the service logic creation offers some advantages, it also has some disadvantages:

Advantages:

- It provides standardized way of service creation.
- The service logic can be user specific and the users may be able to create their own service logic scripts.
- If the network server is a CPL server, the service logic creation does not require any server code modification.
- The service creation and deployment is easy and fast.

Disadvantages:

- The service logic and user data are implemented in one script. This does not give the option for the implementation of a subscriber common service logic and subscriber specific service data.
- There are services, which require loops, function invocation or dynamic service data, e.g. Hunt Group, and Call Distribution.

To illustrate the above, let us consider the Call Distribution service. Suppose that a user has defined the following Call Distribution rules:

- The calls to the users are distributed between two numbers.
- The maximum calls that a number is allowed to answer are 50 per day.
If the number of calls is more than 50 then the calls are forwarded to voice mail.

This service requires management of service data external to the script. This could be achieved with function invocation, which CPL does not allow.

- CPL does not address the service interaction.

4.2.2 Proprietary service logic creation

The service logic is implemented on the SIP server using a proprietary approach. The code of the SIP server is modified, so it can execute the service logic. In this case the service logic can be implemented only by the equipment supplier. The proprietary approach also has its advantages and disadvantages.

Advantages:

- It can handle the complex services implementation and service interaction, by not having the CPL safety limitations.

Disadvantages:

- The service creation and deployment is slow.
- The service logic is equipment supplier specific and not user customizable.

In order to get the full picture of the service implementation issue we must look at the possible network scenarios.

4.3 Network Scenarios

The fact that a user is subscribed to set of services requires the existence of three basic components:

- Service logic of the services the user is subscribed for. The service logic is described above.
- Service data – the user or network information (forwarding number, time, etc.) required for the proper functioning of the service.
- User profile – a list of services or list of conditions and the corresponding services that they are able to trigger for the user.

4.3.1 Potential Scenarios

Depending on where and how these three components are distributed in the SIP environment, we can differentiate the following scenarios:

First scenario: *The service logic service data and user profile (triggers) reside on the end user system and are managed by the user.*

The users contact each other directly and the service is performed purely from end to end. In this case, the network does not participate in the service related operations.

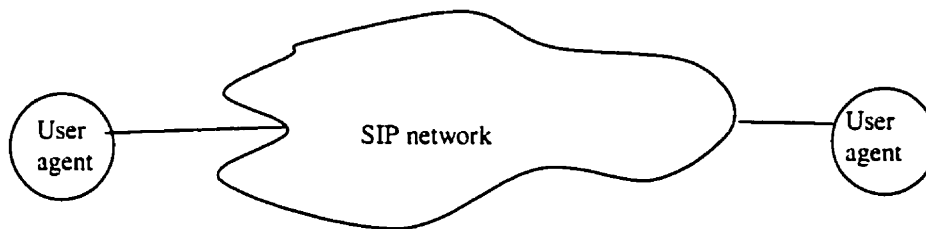


Fig. 4.1

Services reside on the end user systems

Service implementation: The services are implemented in the user agent and the implementation is vendor specific.

Second scenario: *The service logic, service data, and user profile (triggers) reside on the same SIP node, which most commonly would be the Service Provider's SIP proxy server.*

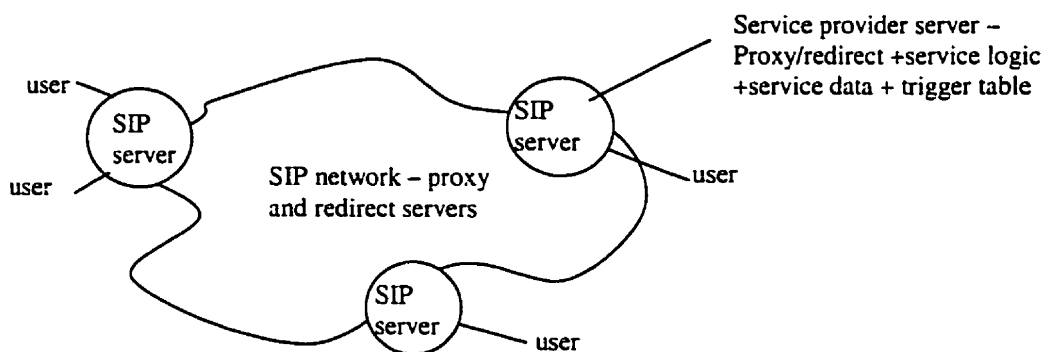
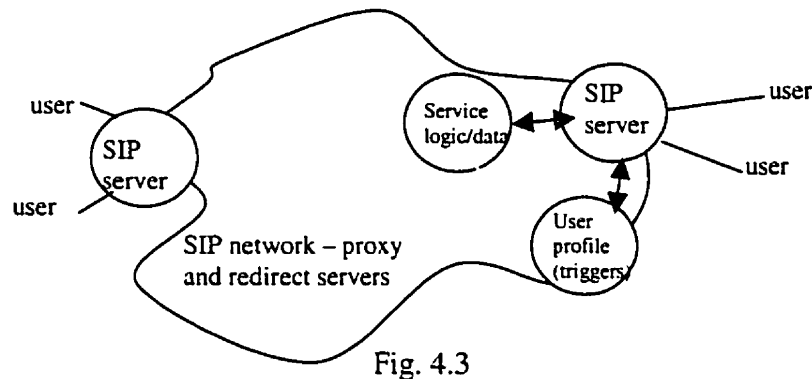


Fig. 4.2

Services reside on a SIP server

Service implementation: The service related operations are performed by one SIP network entity.

Third scenario: *The service logic, data and user profile reside on separate network nodes.* The Service Provider SIP server is the user's entry point to the SIP network. It communicates with the nodes where the service data, logic and user profiles are stored.



Services reside on specialized network nodes

Service implementation: The service related operations are performed in the SIP network.

4.3.2 Scenarios analysis

All the scenarios described above have some advantages and disadvantages.

- Let us start with the first scenario, where the services are implemented in the end system (in the user agent client/server).

The advantage it offers is that a modification of the SIP network is not needed.

The disadvantages are numerous:

- The user agent provides all the functionality and will be too heavy weight software for an end user system.
- In order to be functional some services will require that the user agent is permanently executing.
- The user will not be billed per service but once for the user agent software.

- The second scenario overcomes the deficiencies of the first scenario. A disadvantage is that all the functionality is transferred to the SIP server - the service logic and user data are stored on the SIP server.
- In the third scenario the functionality assumed by the SIP server providing the services is distributed in the network. This simplifies service deployment in the existing SIP network and the service management. Once modified to provide the interface to the service specialized nodes, the SIP network signaling nodes become independent of the service management and provisioning.

In conclusion, let us generalize what are the advantages and the disadvantages of the purely SIP based approach to implementing Internet Telephony VAS.

The most obvious advantage of implementing VAS in SIP networks, using SIP signaling and CPL, is the simplicity of the implementation. Both SIP and CPL are designed to be simple, self-describing, readable and easily extensible. In a purely SIP environment, where only SIP and CPL are used, the VAS implementation can be done with much less efforts than in the more heavy-weight protocol environments as IN/WIN.

The purely SIP approach to VAS inherits the weak points of SIP and CPL. Some of them originate from the fact that SIP is a new protocol. Although SIP provides tools for constructing services, the work on SIP is focused more on getting operational the basic call services - call setup and termination. No SIP network providing the basic call services is commercially deployed, and the development of purely SIP based VAS would require time and resource investment. On the other hand CPL is limited in power - it does not offer means for creation of some more complex IN services and cannot handle the service interaction.

4.4 Building Hunt Group service in SIP environment

After presenting the theoretical issues on the Internet Telephony VAS implementation using SIP, we will focus on one specific VAS – the CMS 8800 Hunt Group service.

4.4.1 Hunt Group service semantics

Hunt Group service allows sequential distribution of the incoming calls to a pilot number. The incoming calls to the pilot number are distributed to a group of members depending on a predefined time schedule. If the called group member is

- inactive/does not answer, or
- busy

then the call is forwarded to the next in the list group member.

The Hunt Group semantics is described in detail below using the following setup:

The user B (pilot number B), subscribed for a Hunt Group service, has defined the following service data:

- 6 groups of phone numbers (Group 1 - Group 6). Each group is assigned one G_default phone number. The incoming call is forwarded to the G_default number if none of the group members can be reached.
- Group time schedule

Group 1	Monday	8.00 am – 12:00 am
Group 2	Tuesday	8.00 a.m. – 12:00 a.m.
	Thursday	8.00 a.m. – 12:00 a.m.
Group 3	Tuesday	1:00 p.m. – 5:00 p.m.
	Friday	1:00 p.m. – 5:00 p.m.
Group 4	Wednesday	8.00 a.m. – 12:00 a.m.
Group 5	Friday	8.00 a.m. – 12:00 a.m.
Group 6	Saturday	8.00 a.m. – 12:00 a.m.

- Default number

The incoming call for B is forwarded to the Default number 765 4321, if no group is scheduled for this time of the day.

The Hunt Group service executed on Monday at 9:00 a.m. is graphically presented in the Fig. 4.4.

Time:

Monday at 9:00 a.m.

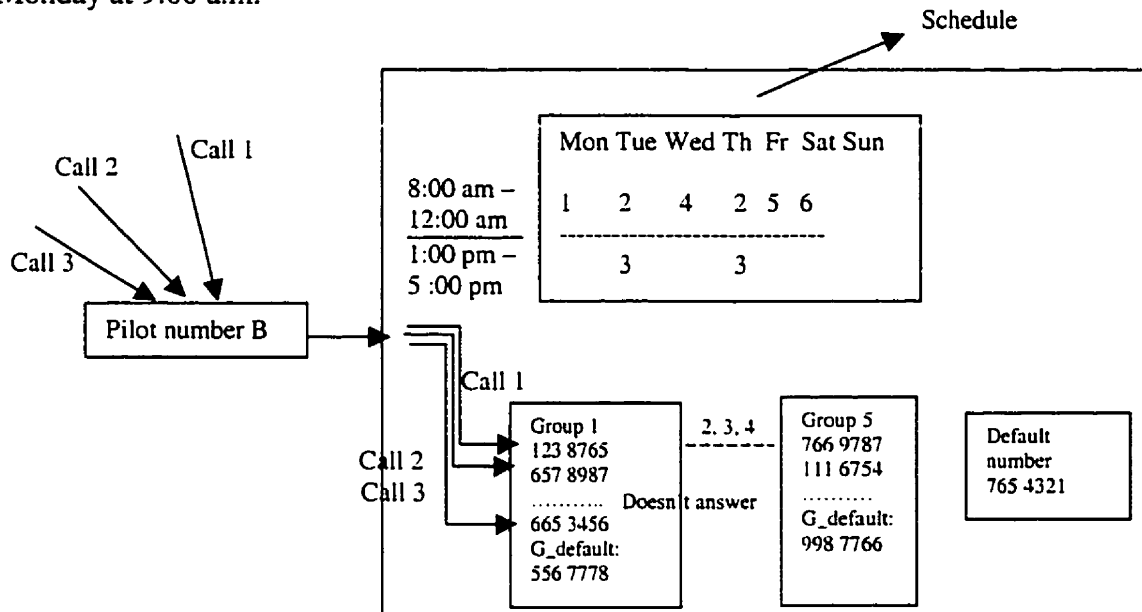


Fig. 4.4

Hunt Group service example

4.4.2 Implementation issues

The following issues must be considered in the service implementation process.

Network scenario: The service logic, service data and user profile reside on the service provider SIP server (i.e., the second scenario described above will be used).

Implementation approach (i.e., CPL or proprietary): A proprietary approach to the service logic implementation should be taken, instead of the CPL approach, for the following reasons:

- There are no existing implementations of CPL enabled SIP servers.

- At the current state of development, CPL does not offer the functionality needed to implement the Hunt Group service logic.

Service data: In addition to the user-defined data described above, the service data include the status of the active group's members, i.e. "contacted" or "non contacted". A new call request must be forwarded only to a "non contacted" group member. If all the members of the active group are already "contacted", the calls are forwarded to the G_default number.

User profile: The user profile may represent a list of the services the user is subscribed for.

Service logic: On receiving an INVITE message for the pilot B number the SIP server starts executing the service logic algorithm presented on Fig. 4.5.

Service Logic Algorithm:

Legend:

A- the caller number

B – the pilot number (the callee number)

G – the group chosen depending on the time schedule.

G_x – a number from the currently chosen group G

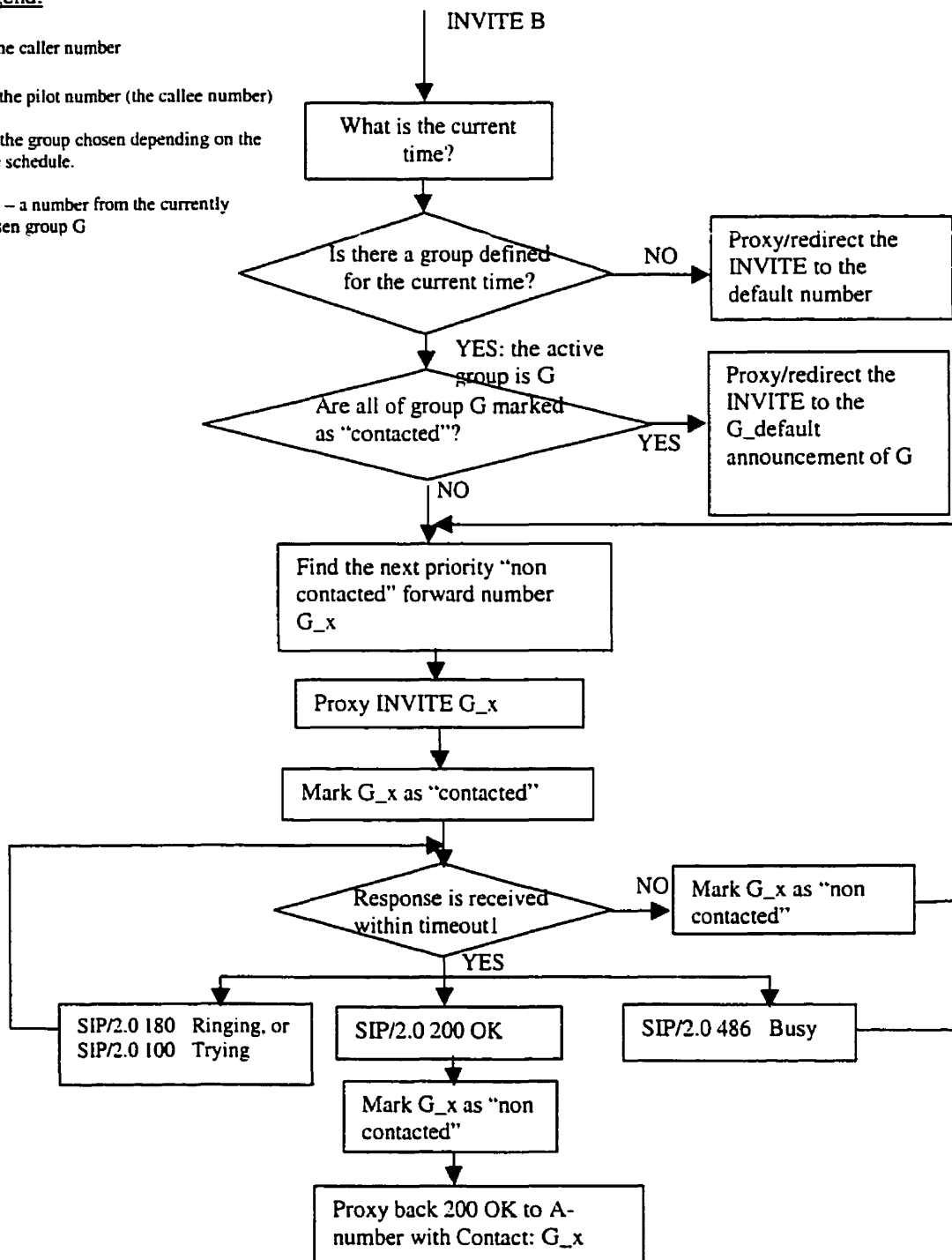


Fig. 4.5

Hunt Group service logic executed when a call to the subscriber B is set up.

Note: We assume that once the G_x user agent has sent an OK response to an INVITE, it responds with “busy” to all subsequent INVITE requests. That is why we can delete the mark “contacted” for G_x .

On receiving a BYE message (Fig. 4.6) from any group member or to the pilot B number, the SIP server transfers the control to the service logic.

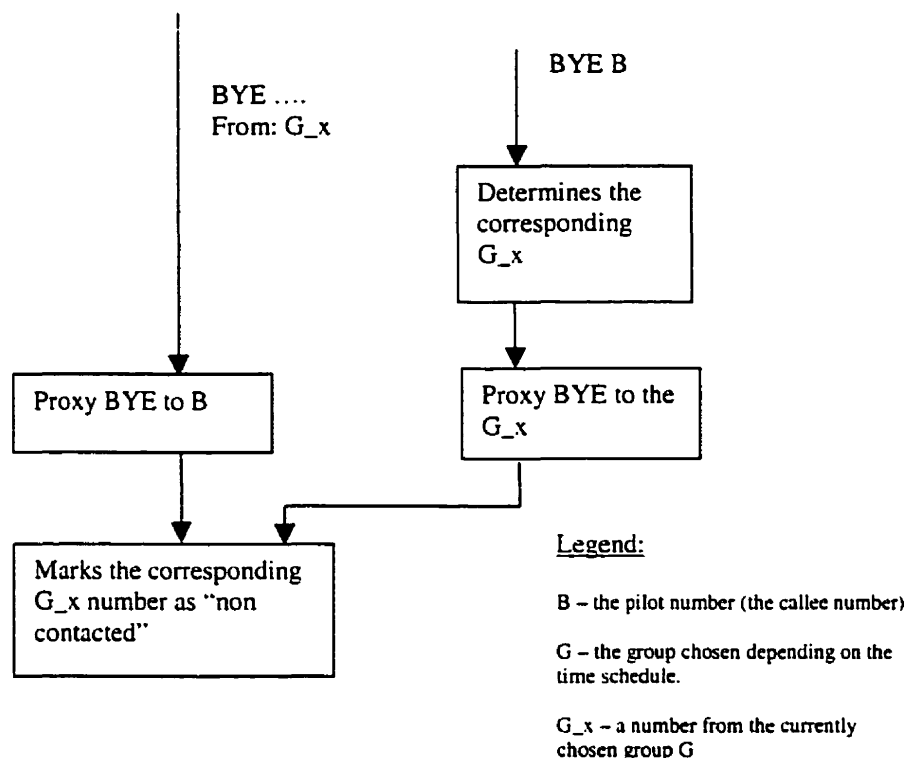


Fig. 4.6

Hunt Group service logic executed when the connection with the subscriber B is terminated

SIP server modification: In order to be able host the Hunt Group service, the service provider's SIP server must be able to :

1. Detect the INVITE request addressed to the pilot B-number.
2. Detect responses sent from the contacted G_x numbers.
3. Detect BYE requests addressed to B or sent from G_x numbers.

The Group Hunt implementation is outlined above to illustrate that proprietary solutions for SIP based implementation of complex VAS are possible. This example does not address the service interaction problem.

4.5 Conclusion

SIP provides the signaling for both the basic (call set up, termination, etc.) and Value Added IP telephony services. SIP also defines the service logic for the basic services. It is built into the SIP architecture entities, which are well specified in the SIP standard. The issue of the service logic creation for the Value Added IP services is addressed by CPL. Although CPL, as it is defined now, is not able to support the Internet implementation of the VAS as they are offered by the Classical Telephony, proprietary solutions based only on SIP are possible. Both CPL and proprietary approach refer to the service creation in purely SIP environment, which implies that VAS are implemented from scratch. This may not be justified for the providers who offer classical telephony VAS and who already have put much investments to build a stable operating IN/WIN VAS infrastructure. This gives the motivation for SIP based VAS implementation that reuses the service logic/data node (SCP) of the IN/WIN architecture.

Chapter 5

Value Added Services implementation using hybrid SIP-IN/WIN approach

As we saw in the previous chapter the Value Added Service logic is the weak point of SIP based IP telephony. In this chapter we present a network scenario for a hybrid SIP-IN/WIN VAS (similar to the third scenario in 4.3, Fig. 4.3) and some of the related implementation issues. Next we give a concise overview of INAP – the communication interface between the nodes in IN environment. It is the basis for studying the problem of the communication interface between the SIP and IN nodes in the presented hybrid network scenario.

5.1 Network scenario

The network scenario for implementing VAS using SIP signaling and IN service logic/data node is given on the Fig. 5.1. The scenario is similar to the third scenario in the previous chapter. But, instead of storing the service logic and data on a SIP network node, it reuses the existing IN/WIN Service Control Points. The service logic and service data reside on the IN/WIN Service Control Point. The user's profiles are stored on a separate server, called the Trigger server. The SCP, as well as the user profiles, are managed by the service operator.

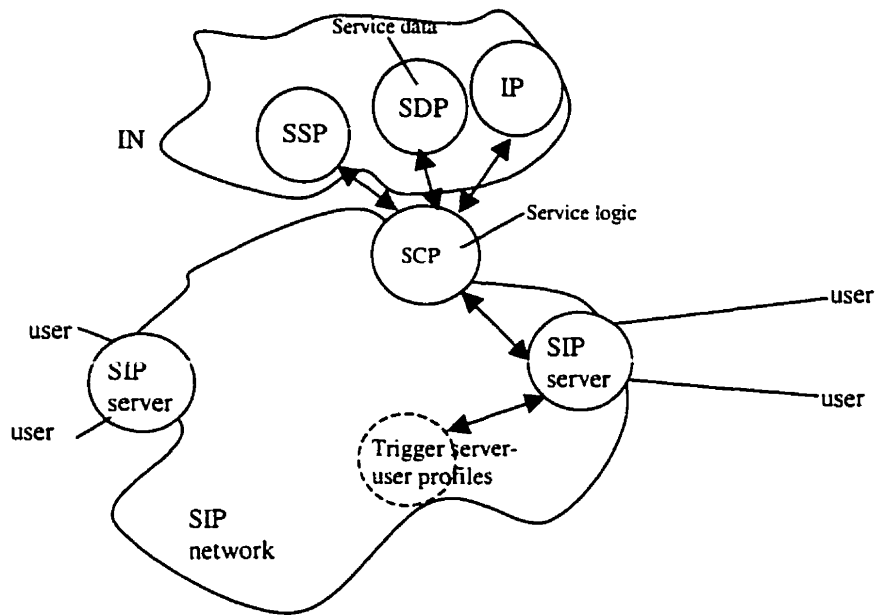


Fig. 5.1

SIP-IN/WIN network scenario

The implementation of Value Added Services in the scenario presented above requires solutions to be found on the following problems:

1. SIP server – SCP communication interface
2. Design solution for the Trigger (user profile) server
3. Interface between the Trigger server and the SIP server
4. Translating the IN specifics (especially detection points) in the terms of SIP session control.
5. Implementing the Call Model in the SIP server.

Further, we will focus on the problem of the communication interface between the SIP server and the Service Control Point. A way to support the IN services with minimal changes on the SCP is to comply the functionality of the SIP server-SCP communication interface to that of the existing protocol for communication between a SSP and SCP in IN environment – the Intelligent Network Application Protocol (INAP). Thus, in the next subsection we focus on INAP.

5.2 Intelligent Network Application Protocol (INAP)

INAP enables the implementation of the services from the IN Capability Sets by supporting the interactions between the following four functional entities: Service Switching Function (SSF), Service Control Function (SCF), Service Data Function (SDF), and Specialized Resource Function (SRF) [Q.1218]. In the common case, these functional entities are implemented in the corresponding physical entities – Service Switching Point (SSP), Service Control Point (SCP), Service Data Point (SDP), and Intelligent Peripheral (IP). Therefore, INAP can be viewed as a protocol providing for the communication between these entities, as shown in the Fig. 5.2.

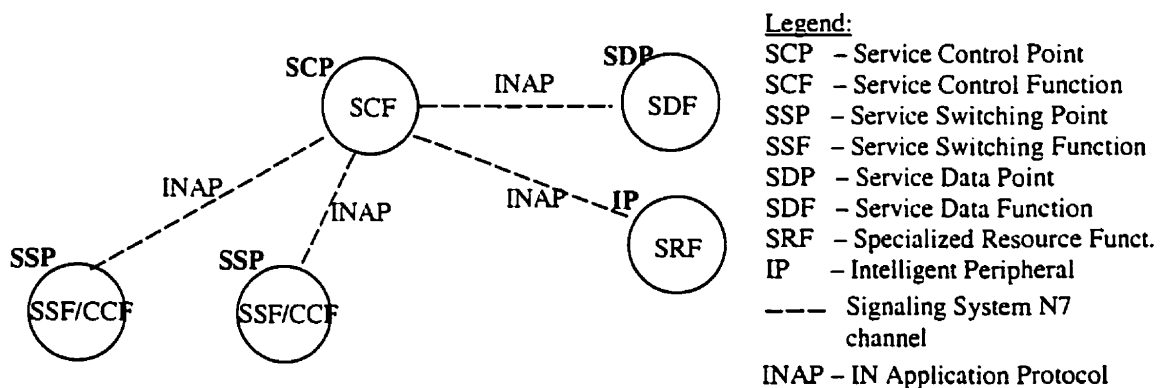


Fig.5.2

INAP – communication interface between IN nodes

It has to be noted that the presented hybrid network scenario reuses not only the IN SCP but also the SDP, and IP. Nevertheless only the SCP communicates directly with SIP network. As in the pure IN environment the SDP and IP are accessed only by the SCP. Since the problem studied involves only the SCP and the SSP, INAP will be presented with focus on the SSP-SCP interface.

5.2.1 INAP environment

The environment of INAP [Q.1208] is an environment specified as Application Layer of the OSI model. The SS7 protocol architecture [MOD90] that INAP is built on and its mapping to the OSI model is illustrated in Fig. 5.3.

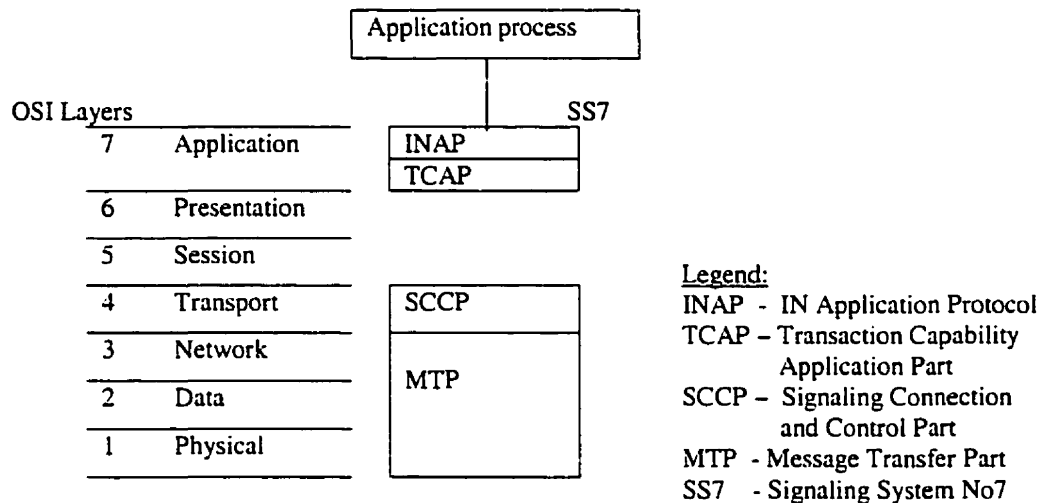


Fig. 5.3

Structure of the SS7 stack

INAP provides its services to the Application process and in its turn uses the Transaction services of the Transaction Capabilities Application Part (TCAP) of SS7. TCAP is supported by the services of the connectionless Signaling Connection Control Part (SCCP), which is built on top of the Message Transfer Part (MTP) protocol.

5.2.2 INAP vocabulary

Generally the INAP vocabulary contains definitions of operations and their parameters, Application Service Elements (ASEs), and Application Contexts (AC) as described next.

Operations

An Application Process establishes an association (i.e. a pipe through which it communicates with another process) [FAY97] and then executes the necessary

operations. The operations implementing the SSF/CCF – SCF interface are carried by ROSE -based TCAP. ROSE stands for Remote Operation Service Element. It provides facilities for remote procedure call, by specifying four Protocol Data Units – *Invoke*, *Return Result*, *Return Error*, and *Reject*. The user issues a request for an INAP operation to be performed via a procedure call corresponding to a ROSE *Invoke* Data Unit. This operation is carried enveloped within the ROSE *Invoke* to the server, at which point it may send back its result using the ROSE *Return Result* Data Unit.[FAY97].

The INAP specification of each operation includes the parameters that can be send, when invoking the operation (ARGUMENT clause), the parameters returned with the result (RESULT clause), and the parameters returned in error cases (ERRORS clause).

Application Service Elements (ASEs)

The ASEs are just group of operations. The operations are grouped based on criteria for functionality (e.g. traffic management), modular reuse and future evolution [MAG96]. INAP makes use of general ASEs as ROSE mentioned above or Transaction Capabilities ASEs (TCs). But it also defines 25 INAP-specific ASEs.

Application Contexts

As INAP aims for modular structure, ASEs can be grouped in so-called Application Contexts (AC), where an AC consists of a combination of ASEs and the relationships between them. An Application Context is typically a subset of the total INAP and specifies that portion of the protocol needed for communication between two types of IN functional entities, e.g. between SSF and SCF [MAG96].

The following graphic summarizes the relationship between the INAP concepts described above.

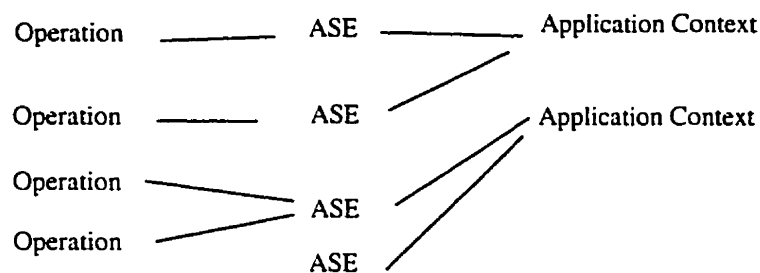
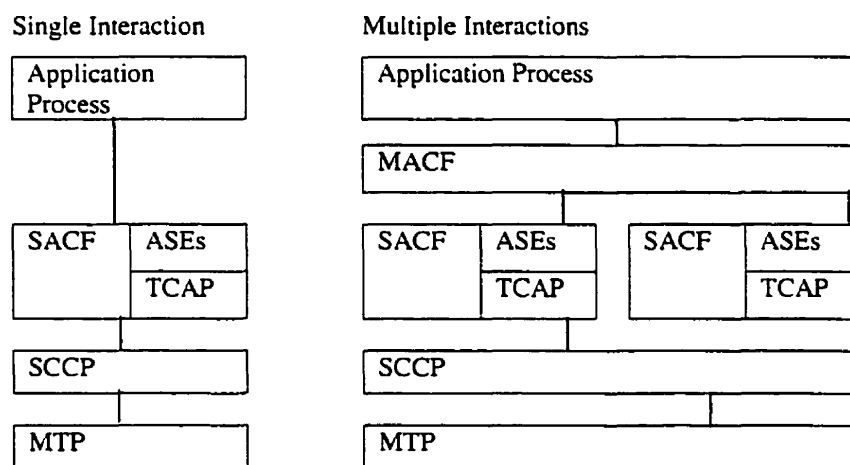


Fig. 5.4

Structure of INAP

5.2.3 INAP architecture

The INAP protocol architecture can be described as follows [MAG96]. A physical entity has either a single interaction or multiple interactions with physical entities. In the case of single interaction, a Single Association Control Function (SACF) provides a coordination function in using a set of Application Service Elements (ASEs). In the case of multiple coordinated interactions, a Multiple Association Control Function (MACF) provides the coordination among several sets of ASEs. The Fig.5.5 [Q.1218] illustrates the INAP architecture.



Legend:

SACF – Single Association Control Function	ASE – Application Service Element
MACF – Multiple Association Control Function	INAP – IN Application Protocol
TCAP – Transaction Capability Application Part	MTP – Message Transfer Part
SCCP – Signaling Connection and Control Part	SS7 – Signaling System No7

Fig. 5.5

INAP protocol architecture

5.2.4 INAP procedures

The procedures describe the sequencing of the protocol operations [FAY97] and they are addressed in Clause 3 of [Q.1218]. In this Recommendation the INAP procedures are defined in a way that fulfil two contradictory requirements. The first one does not constrain the service logic developers, and for this purpose the INAP procedures prescribe the sequencing of the operations rather loosely. The second one is to ensure interoperability between the products from different vendors. This is achieved through a constraining of the operation sequencing.

Therefore when implementing a service logic, the provider has some freedom in interpreting the procedures in relation to specific services and service features, but their products must be compliant to the INAP procedures specification.

5.2.5 INAP encoding

The INAP operations are specified using the Abstract Syntax Notation 1 (ASN.1) formalism [Q.1218]. ASN.1 is Pascal-like language for describing data in terms of a set of application specific data types [FAY97], [NEU92]. It is a formal tool that permits the precise definition of types and information exchanged. It is associated with a set of rules that specifies the format of the values of each of the types called “encoding rules”. Existing sets of encoding rules for ASN.1 are BER (Basic Encoding Rules) [X.690] and PER (Packet Encoding Rules) [X.691]. BER is used for encoding of the ASN.1-specified INAP operations.

5.3 Conclusion

The concept of hybrid SIP – IN VAS is promising in terms of reuse of the strong existing IN base for building Internet Telephony services. However, before being tested in practice, it would require a number of design problems to be solved. As in the pure IN

environment, a key issue is the communication interface between the network nodes, and it will be studied in the next chapter.

Chapter 6

Lightweight SIP protocol for SCP - SIP server communication

After presenting the basic concepts of the INAP, let us go back to the suggested hybrid SIP-IN scenario for implementing VAS and, more specifically, to the problem of the communication interface between the SIP network server and the IN Service Control Point. There are several possible approaches to defining a SIP-SCP communication interface. This chapter presents a lightweight protocol for SIP-SCP communication, build as extension to SIP. Then it describes the protocol behavior of the communicating SIP and IN nodes.

6.1 SCP – SIP server interface

Below are presented two general ways to address the problem of the communication between a SCP and a SIP server:

- API based approach

Two alternatives, which can be mentioned, are CORBA and Java/RMI. In this case a CORBA or Java/RMI server object is implemented on a SCP and interacts with other CORBA or Java/RMI remote client objects, implemented on the SIP servers. The CORBA or Java server has an interface and exposes a set of methods, implementing the set of INAP operations, to the remote client object. To request a service a CORBA or Java client acquires a reference to an object server and makes method calls as if the server object resides in the clients address space [RAJ98].

- Message based (protocol) approach.

Here we also will mention two ways of providing for the INAP functionality.

- Carrying INAP over IP

The INAP data are encoded using the BER encoding scheme and are sent over IP/UDP or IP/TCP connection.

- Defining an ASCII IP protocol supporting INAP

The INAP messages are encoded as a text and are carried over IP/TCP or TCP/UDP as most current Internet application protocols – HTTP, SMTP, ftp, etc.

When considering the above alternatives we have to take into account that one of the communicating entities (notably the SIP server) is an Internet node and building the communication interface as an Internet application layer protocol is a natural choice. Therefore the arguments for the format of the communication protocol between the SIP server and the SCP are the same as the arguments for choosing the format of any Internet application layer protocol. Following the analysis given in [SIT98] we can summarize this arguments as follows:

- Using systems like CORBA or Java/RMI has not been widely used for Internet application protocols due to the fact that they are relatively immature, and they lack interoperability and widespread cross-platform availability.
- Using ASN.1 and BER encoding rules are also not widely used in the Internet. The parsing is cumbersome and requires parsers for both ASN.1 and BER.

Considering the above we can see that the text based protocol approach is the most promising in terms of simplicity, and reuse of the existing Internet application protocols implementations. It gives the motivation for the next sections, which are devoted to the development of an ASCII IP based SIP server-SCP interface. They present the design of a lightweight protocol based on the Session Initiation Protocol, defined as SIP extension “inap.sip”.

6.2 SIP extension – inap.sip

This section will define an extension to Session Initiation Protocol - inap.sip. The purpose of the inap.sip extension is to provide tools for the SIP servers to access the Service Logic stored on IN based Service Control Points (SCPs), and thus to bring to the Internet Telephony the Value Added Services developed in the existing Intelligent

Networks. Figuratively, the inap.sip extension will teach the SIP servers to talk INAP - the language spoken between the IN SSPs and SCP.

For brevity in the next sections, SIP extended with the inap.sip extension will be referred to as SIPext. According to the SIP extension mechanism, when using the extensions described in inap.sip, the SIPext clients must include the extension name in a *Require:* header.

Before going into the description of the inap.sip, let us first present the architectural entities participating in the SIP-IN VAS scenario.

6.2.1 Architecture entities

An example network configuration for the SIP-IN scenario is presented in the Fig. 6.1

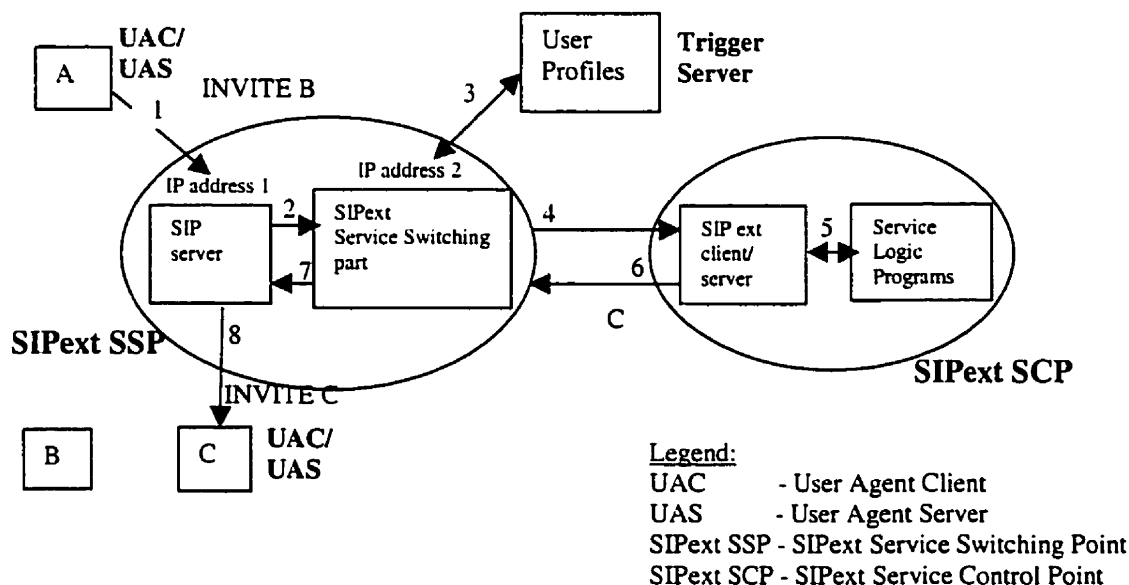


Fig. 6.1

SIP/IN network configuration

The network configuration on Fig. 6.1 comprises the following architecture entities:

User Agent Client/Servers

The user agent clients and user agent servers are pure SIP entities and were described in Section 3.1.

Trigger Server

The Trigger server stores the user profiles, i.e. the list of armed triggers. In SIP terms the triggers represent specific SIP messages and the values of specific message parameters, which once encountered, trigger a service execution. The list of armed triggers gives the match between a trigger and the INAP operation uniquely identifying the service that has to be executed.

SIPext Service Switching Points (SIPext SSPs)

The SIPext SSP is composed of two functional parts: a SIP server part and the SIPext Service Switching part.

- The SIP server part operates as a pure SIP proxy server and provides for the basic call service, i.e. call setup and call termination.
- The SIPext Service Switching part is a SIPext server/client, which plays the role of an IN Service Switching Point. It provides for:
 - trigger detection;
 - sending SIPext requests to the SCP that trigger Service Logic Program (SLP) execution. It also can send requests and responses for interaction with an already running SLP;
 - receiving and interpreting the SIPext responses or requests from the SCP and
 - transferring the control to the SIP server part of the SIPext SSP for performing the corresponding SIP signaling actions.

As every node connected to the Internet, the SIPext Service Switching Point must be identified with at least one IP address. Before considering the design options of assigning one or two IP addresses to it, we first have to stress that the SIPext Service Switching Point is specific in the sense that it have dual functionality. From one side it operates as a pure SIP proxy server and from another side it operates as a SIPext client/server.

Therefore we have to be able to classify the messages so those related to the basic call setup are processed by the SIP proxy and those related to the services execution are processed by the SIPext part. Depending on whether we assign one or two IP addresses to the SIPext Service Switching Point we can achieve a separation of the pure SIP from the SIPext messages in three possible ways:

- The same IP address and ports (UDP and TCP) are assigned to the SIPext Service Switching Point. In this case all SIP and SIPext messages are received by the SIP proxy. Then they are classified depending on the *Require:* header. If the *Require:* header specifies inap.sip extension, the message is classified as a SIPext message and is transferred to the SIPext part for processing. All other messages are processed further by the SIP proxy.
- The IP address is unique, but the port assigned to the SIP proxy (e.g. the default SIP port 5060) and the port assigned to the SIPext client/server are different. This way the pure SIP messages will be received by the SIP proxy and SIPext messages will be received by the SIPext client/server. This will reduce the workload of the SIP proxy. It also will simplify the implementation, because classifying the messages by their *Require:* header will not be necessary.
- The SIPext Server Switching Point has two IP addresses – one is assigned to the SIP proxy, the second one to the SIPext client/server. In addition to the natural classification of the received messages, this allows assigning to the SIPext part the default SIP port. Although using a default SIP port is not mandatory it facilitates the SIPext SCP operation when sending requests to the SIPext SSPs and ensures interoperability.

As result a of the above considerations we base our further study on the case where the pure SIP and the SIPext Service Switching part have different IP addresses. In the following diagrams the IP address “provider.com” corresponds to the pure SIP server and the address “ssp.provider.com” corresponds to the SIPext client/server.

We start our study by presenting the basic operation of the SIPext nodes.

SIPext Service Control Points (SIPext SCPs)

The SIPext Service Control Point also combines the functionality of SIPext server/client and IN SCP.

- The SIPext server/client part:
 - provides for the communication with the SIPext SSP;
 - interprets SIPext messages;
 - transfer the control to the IN Service Logic Programs (i.e. to the IN SCP part of the SIPext SCP);
 - formulates SIPext messages.

6.2.2 SIPext basic operation

The operation of the SIPext is described using the configuration from the Fig. 6.2 in the following call context:

The user A places a call to the user B. The user B is subscribed with the provider.com for some service. The user B profile is stored on the Trigger server. Since the SIPext operation does not relate to the interaction between the Trigger server and the SIP SSP, the Trigger server is omitted from the diagram. The Fig 6.2 summarizes the basic SIPext operation and is explained in detail below.

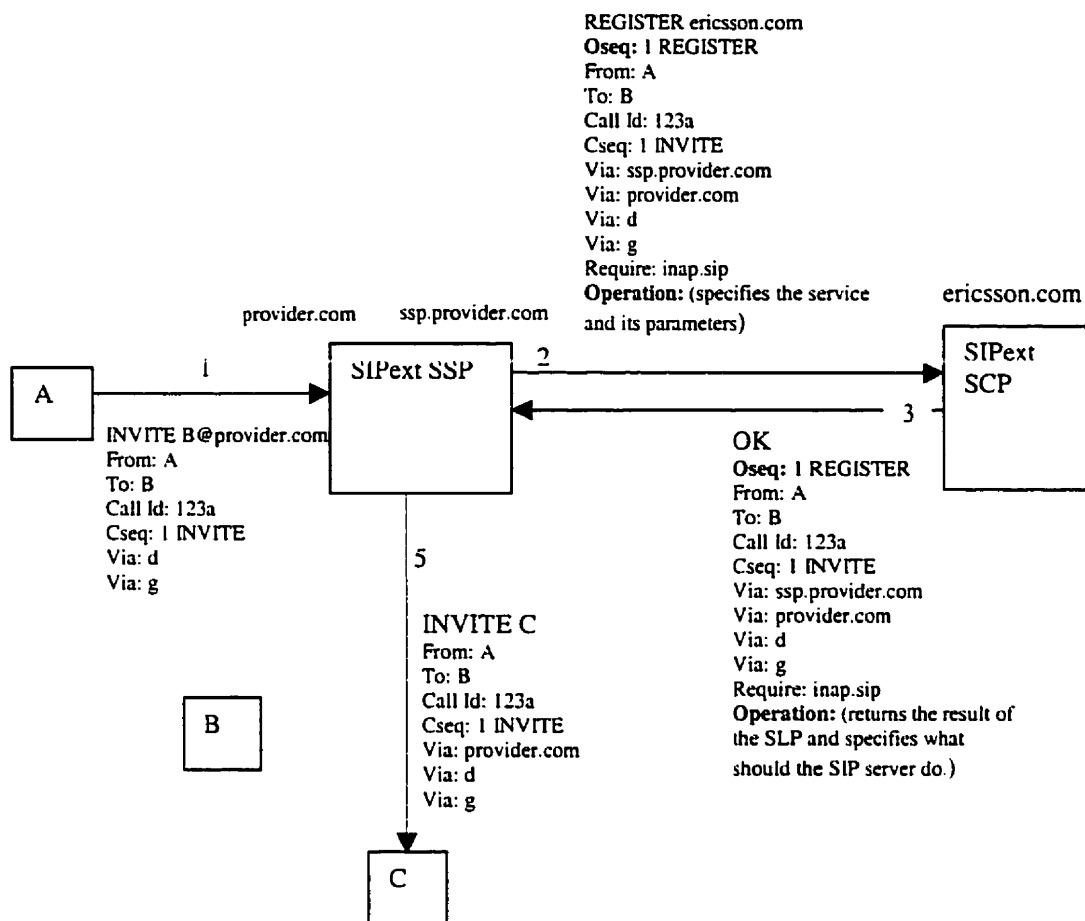


Fig. 6.2

SIPext basic operation

When a message for a user subscribed with the service provider arrives, the SIPext SSP server consults the user profile stored on a Trigger server (not shown on the Fig.6.2). If the user is subscribed for a service, it formulates a SIPext request message and sends it to the SIPext SCP. The SIPext messages represent SIP REGISTER requests and responses that contain headers defined in inap.sip.

The choice of REGISTER as the request carrying the requested INAP operation is motivated by the following:

- From the request methods defined in SIP we definitely exclude INVITE, which is the only SIP request that requires acknowledgment.
- The rest - BYE, CANCEL and ACK, are mostly related to the call termination. The OPTIONS method is related to querying the user capabilities and status.

- REGISTER is the SIP request method that is the most service oriented and in this sense it seems the most appropriate choice.

The SIPext REGISTER request message triggering a service will contain a SIPext header field specifying the Service Logic Program (SLP) to be executed by the SCP.

On receiving the REGISTER request, the SIPext SCP examines the SIPext header fields and eventually starts the corresponding Service Logic Program. The result of the successful SLP execution is returned to the SIPext SSP in an OK response message.

On receiving an OK response from the SIPext SCP, the SIPext SSP analyzes the SIPext header fields returned within the response message. Depending on the content of the header fields it may proceed either by generating a pure SIP message to specific address and sending it over the SIP network, or perform some monitory and notifying actions.

The basic operation of the SIPext SSP and SIPext SCP was described above in the case of a successful execution of the SLP and as consequence generation of an OK SIP response message. It is possible that the SLP does not terminate successfully and an error message is sent to the SIPext SSP. The SIPext SSP must be able to determine whether it should reformulate and return an error response to the caller or it should perform some other actions.

The error responses will be considered in more detail in the description of the behavior of the SIPext Service Switching part and the SIPext SCP.

6.2.3 Definition of inap.sip

In this section the proposed SIP extension inap.sip will be defined. The purpose of the inap.sip extension is to allow the use of SIP as a protocol for communication between SIP servers and IN based Service Control Points and will represent an Internet implementation of INAP. The SIP extension must be able to offer the same functionality as the IN Application Protocol.

The proposed inap.sip extension consists of the four header fields: “*Operation:*” “*Oseq:*” “*Result-Op:*” and “*Error-Op:*”. They are described in Augmented Backus-Naur Form (BNF). The constructs and basic rules of BNF are presented in [rfc1945].

6.2.3.1 *Operation:* header field

The *Operation:* header field advises the recipient to perform the operation specified in the content of the header. Except the name of the operation, the header can contain the parameters of the operation (if any). The operation, the parameters’ name, type and value correspond to those defined in INAP.

The format of the *Operation:* header field is given below.

```
Operation = "Operation" ":" name-operation ";" [operation-parameters]
name-operation = 1*alpha
operation-parameters = *("parameter" ":" parameter-value";")
parameter = *((1*alpha))
parameter-value = *DIGIT|*CHAR|boolean|*OCTET
```

It has to be stressed that INAP is very powerful but also rather heavy protocol providing for a large variety of VAS. It is specified using the ASN.1 notation. ASN.1 is well suited for the description of the INAP operations parameters, which are represented by nested data structures with a lot of optional elements.

On the other hand the text-based parameter-value structure of the Internet application protocols, including SIP, works well where the parameters are not structured and values are simple lists [SIT98].

Before studying the problem of the text based representation of the INAP operations let us first look at the specifics of the data structures representing INAP operations arguments.

The operation arguments can be classified in two categories:

- The operation argument represents one dimensional tree data structure like the instance of the InitialDp operation argument given below (Fig. 6.3).

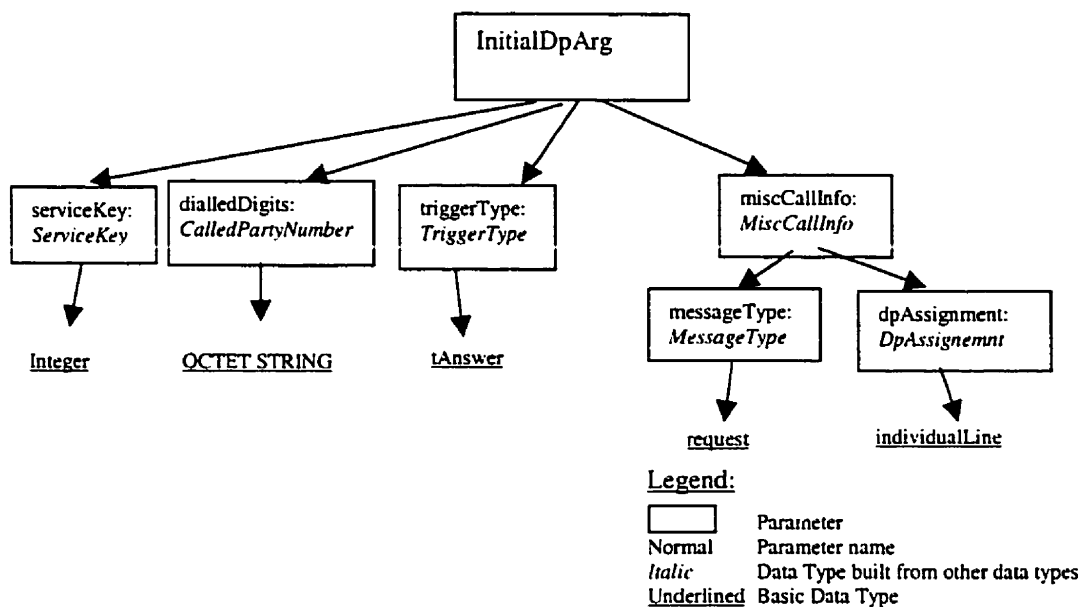


Fig. 6.3

Argument structure of the InitialDp operation

- The operation argument represents multidimensional tree data structure. An example is the following instance of the CallInformationReport argument (Fig. 6.4).

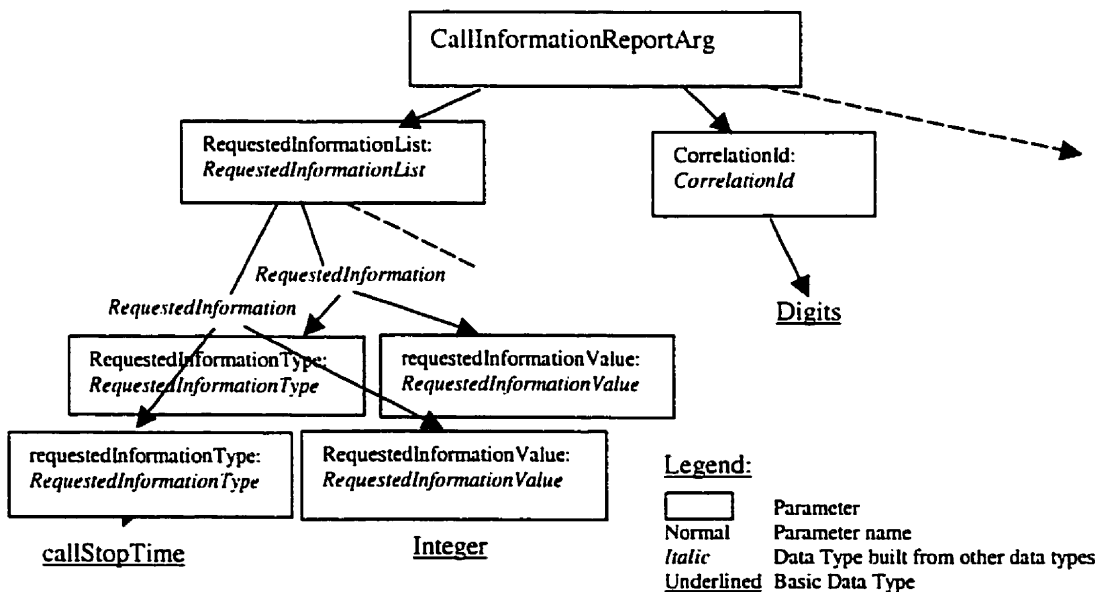


Fig. 6.4

Argument structure of the CallInformationReport operation

In either case there are two possible ways to describe linearly the argument data:

1. The basic type data (i.e. the leaves in the tree) are identified with the full path of data types that are above them in the hierarchy of the argument data structure. This path explicitly describes their location in the argument data structure tree. The receiving entity must have knowledge about the corresponding argument data structure and the data types that it is built of.

In this case the InitialDP operation parameters will be represented by the following sequence:

```
/serviceKey:40, /dialledDigits:8876hjpgda, /triggerType:oAnswer,  
/miscCallInfo/messageType:request, /miscCallInfo/dpAssignment:individualLine.
```

The CallInformationReport parameters will be described by the sequence:

```
/requestedInformationList/requestedInformationType: callStopTime,  
/requestedInformationList/requestedInformationValue:457,  
/requestedInformationList/requestedInformationType: callElapsedTime,  
/requestedInformationList/requestedInformationValue: 20,  
/correlationId: 111.
```

2. Each basic type data is identified with one parameter name.

In the example of the InitialDP operation the parameters can be described with the following names:

```
/serviceKey:40, /dialledDigits: 8876hjpgda, /triggerType:oAnswer,  
/messageType:request, /dpAssignment:individualLine.
```

In the example of the CallInformationReport operation the parameters may be given as the following sequence:

```
/requestedInformationType:callStopTime, /requestedInformationValue:457,  
/requestedInformationType:callElapsedTime, /requestedInformationValue: 20,  
/correlationId: 111.
```

This approach imposes that, in addition to its knowledge on the argument data structure and its building data types, the entity interpreting the operation must be able to match the parameter name to a unique location (field) in this data structure for storing the data.

Before concluding which of the two approaches is more feasible for the SIPext implementation several aspects must be considered.

One of them is the space efficiency of the message format, because with the application Internet protocols it is desirable to avoid the 1500 byte UDP packet fragmentation limit. The text-based format Internet protocols are in general less space efficient than ASN.1 or Internet binary-format protocols [SIT98]. The space efficiency is even more of concern when using a text-based protocol for transporting the INAP operation parameters. In order to benefit from the advantages of the textual encoding (simple text processing, readability, simplified debugging, manual entry) in the current work the parameter and data type names are kept as they are defined in INAP and as consequence they are rather long. If the parameter data type paths are included in the message (second approach) it is highly possible that the size of the message will exceed the 1500 bytes fragmentation limit.

The problem of the parameter data type presentation (first or second approach) has also to be considered in the aspect of implementation complexity and execution time.

The lightweight Internet text-based implementation of INAP described here does not aim to evaluate comparatively the feasibility of the two approaches.

The *inap.sip Operation*: header format as defined above allows either way of operation parameters presentation. For the purpose of the following examples it is assumed that the basic type data parameters are uniquely identified by the parameter name, i.e. the first approach to the parameter representation is used.

Examples:

Operation: InitialDP; /eventTypeBCSM:collectedInfo;

Operation: CallInformationRequest;
/requestedInformationType: callStopTime;
/requestedInformationType:callAttemptElapsedTime;
/requestedInformationType:releaseCause;

Operation: CallInformationReport;
/requestedInformationType: callStopTime; /requestedInformationValue:457;
/requestedInformationType:callAttemptElapsedTime; /requestedInformationValue:20;
/requestedInformationType:releaseCause; /requestedInformationValue:hdgjdfjd;

Operation: RequestReportBCSMEvent; /eventTypeBCSM: oAnswer;
/monitorMode:notifyAndContinue; /eventTypeBCSM:oAbandon;
/monitorMode:notfyAndContinue;

6.2.3.2 *Result-Op*: header field

The *Result-Op*: header field carries the result of the successful execution of an operation. This header is present in the response message only if the operation requires that the result of the operation is sent back to the entity (SSP or SCP) that invoked the operation.

The format of the *Result-Op*: header field is:

Result-Op = "Result-Op"" : " [result-arguments]
result-arguments = *("argument" ":" argument-value ";")
argument = */(1*alpha))
argument-value = *DIGIT|*CHAR|boolean|*OCTET

Example:

Result-Op: /digitsResponse: 456;

This is the result returned after the execution of ReceivedInformation operation and represents an instance of the ReceivedInformationArg.

6.2.3.3 *Error-Op*: header field

The *Error-Op*: header is used to convey information about the reason for the unsuccessful operation execution.

The format of the *Error-Op*: header is:

Error-Op = "Error-Op" error-name ";" [(error-parameters [":" parameter-value]) ";"]

error-name = 1*alpha

error-parameters = *(1*alpha)

parameter-value *DIGIT|(1*alpha)

Examples:

Error-Op: CancelFailed; problem:unknownOperation; operation:654;

Error-Op: MissingParameter;

6.2.3.4 *Oseq*: header field

The SIPext node that generates a SIPext REGISTER request adds the *Oseq*: header field to every request. This number denotes the order of the operation requests sent for a specific *Call-Id* and imposes that the operations are executed in the same order. The *Oseq*: sequence number is also used to match the responses to the requests.

The format of the *Oseq*: header field is:

Oseq = "Oseq" 1*DIGIT "REGISTER"

Note: The REGISTER literal is used, because the REGISTER method is the only one used for SIPext requests.

Example:

Oseq: 3 REGISTER

6.2.4 Behavior of the SIPext SCP

The SIPext SCP may act as a server or as a client. In the common case the SIPext SCP operates as a server. In this case it receives the REGISTER requests sent by the SIPext Service Switching part, starts the execution of the INAP operation specified in the request, and returns the corresponding response.

It is also possible that the SIPext SCP acts as a client, i.e. it issues a REGISTER request specifying some operation(s) to be executed by the SIPext Service Switching part. This can be needed for the implementation of some IN specific VAS, such as services involving prompting and collecting additional information from the caller, filtering calls (used in televoting or mass calling services), etc.

6.2.4.1 SIPext SCP acting as a server

On receiving a REGISTER request (see Fig. 6.2), the SIPext SCP proceeds as follows:

- Checks whether the REGISTER request comes from a SIPext SSP (the topmost *Via:* header field), which is authorized to access it.
- Using the *Oseq:* sequence number it determines whether the request has to be processed next. The request will be postponed if there are any requests with smaller numbers that have not been received and processed yet. The server must remember the number of the last request processed for every *Call-Id*.
- Searches for *Operation:* header fields. All other headers (pure SIP headers which are no service related) and their content are kept as they were received.
- The *Operation:* header field is parsed. Depending on the name of the operation the corresponding INAP operation is performed by the SCP. For example a Service Logic Program is started.
- If the operation has been executed successfully an OK response is generated. The OK response may contain another *Operation:* header field or *Result:* header field. The no service related header fields (pure SIP header fields) are copied in the OK response as they were in the INVITE request.

- If the operation hasn't been executed successfully an INAP error is returned. The SIP SCP generates a SIP error response 400 (Request Failure) or 500 (Server Failure). An *Error-Op:* header is included in the response specifying in INAP error and parameters. The table below gives examples of INAP errors and the corresponding SIP error response message used to transport it to the SIPext SSP.

INAP ERRORS	SIP error responses
MissingParameter	400 Bad Request
ParameterOutOfRange	400
SystemFailure	500 Server Internal Error
MissingCustomerRecord	500
TaskRefused	400
UnexpectedComponentSequence	400
UnexpectedDataValue	400
UnexpectedParameter	400
RequestedInfoError	400

Table 6.1
INAP errors transport

Although the type of error can be related to the type of SIP error message (400 Bad Request or 500 Server Internal Error) there are some INAP errors that does not fit in any SIP error class. This is not of much importance since the complete information for the INAP error and its parameters is carried in the *Error:* header.

The operation of the SIPext SCP described in the text above is summarized with the pseudo-algorithm on Fig. 6.5.

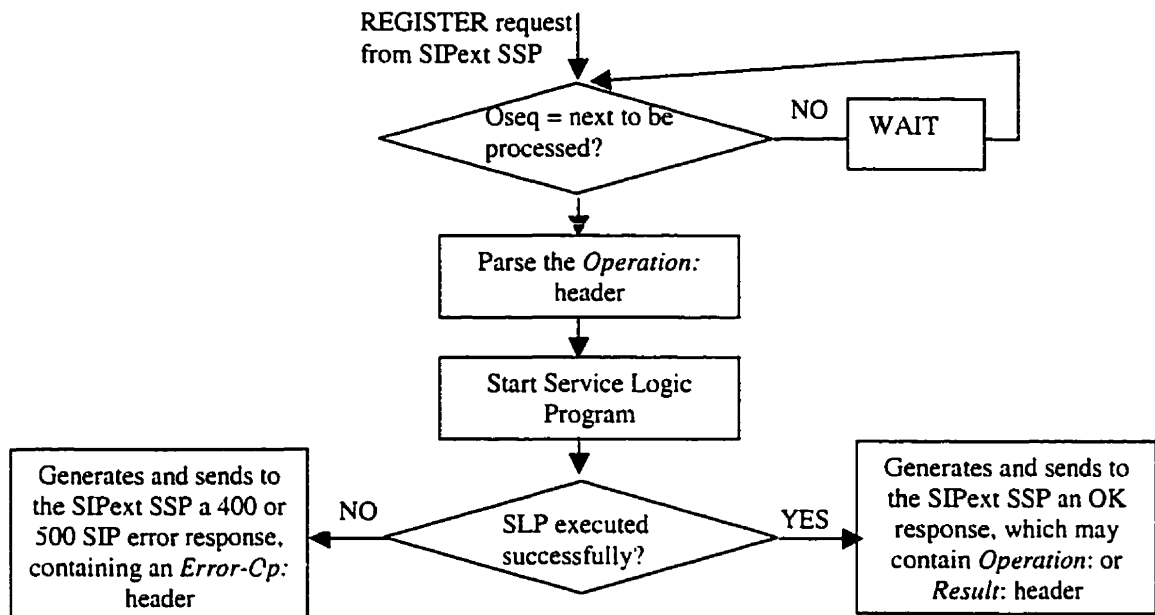


Fig. 6.5

SIPext SCP acting as a server

6.2.4.2 SIPext SCP acting as a client

It is possible that the SIPext SCP requests that the SIPext SSP executes an operation by sending to it a REGISTER request. This case is graphically presented in the Fig. 6.6.

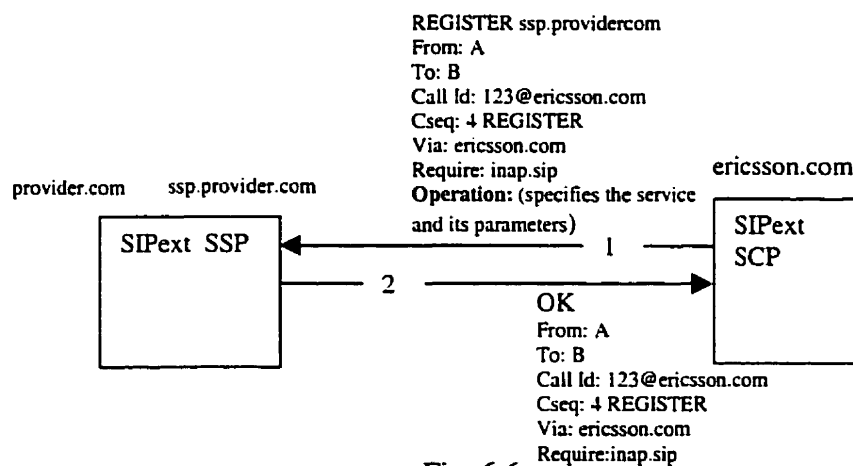


Fig. 6.6

The SIPext SCP acting as a client (sending a REGISTER request)

Generally a REGISTER request sent by the SIPext SCP requires the execution of operations which are not related to the call setup. In this case they carry much less information than the REGISTER request sent by the Service Switching part. The requests sent by the SIPext SCP are used to instruct the Service Switching server to perform some call monitoring and event notification actions.

When generating the REGISTER the SIPext SCP uses the corresponding call context data to fill the required pure SIP headers *To: From:, Cseq:*. Generally these headers are not used by the Service Switching server, because in most cases the information they carry overlap with the data contained in the *Operation:* header.

After sending the REGISTER request the SIPext SCP server proceeds as follows:

- Waits for a response message from the SIPext SSP.
- Processes the response message. If either the *Result-Op:* or *Error-Op:* header is present, it is analyzed by the server and the result or error parameters (if any) are transferred to the Service Control Point for processing.

6.2.5 Behavior of the SIPext Service Switching Point

As shown in the Fig. 6.1, the SIPext SSP consists of two functional parts – a pure SIP part, which performs the basic call setup/termination signaling, and SIPext Service Switching part, taking over the signaling communication with the SIPext SCP. In this section, the behavior of the SIPext SS part will be presented.

Similar to the SIPext SCP, the SIPext Service Switching part functions either as a client or as a server.

6.2.5.1 SIPext Service Switching part acting as a client

In the general case the SIPext SS part is the client in the SIPext SSP – SIPext SCP communication. The Service Switching part sends requests to the SCP either when a trigger for a service is detected, or an event has to be reported to the SCP. In order to be

able to determine when a REGISTER request has to be sent to the SIPext SCP, the Service Switching part must monitor the SIP messages received by the pure SIP server.

A. REGISTER request is sent in case a trigger for a service is detected.

In order to be able to detect a trigger for a service, the SIPext Service Switching part must monitor all the INVITE requests sent to the pure SIP proxy server (in the example the SIP proxy server has address *provider.com*). For each INVITE it consults the Trigger server. If the user is subscribed for a service the SIPext Service Switching part generates and sends to the SCP a REGISTER request for the service execution.

The Service Switching part formulates the REGISTER request in the following way (see Fig.6.2):

- Copies all the headers as they were in the INVITE request. This allows the Service Switching part to operate in stateless mode. It formulates the REGISTER request to the SCP and forgets the call state. It doesn't need to store call information while waiting for response from the SCP, because it is carried in the SIPext messages.
- Adds a *Via:* header field containing the address of the Service Switching server.
- Adds *Oseq:* header containing the sequence number of the REGISTER request. This number is used by the SS server to match the responses. It is also used by the SCP to execute the requested operations in order.
- Adds one or more *Operation:* headers, specifying the operation to be executed by the SCP and its parameters.

B. REGISTER request is sent as result of event detection.

Some services require that the Service Switching part reports to the SCP the occurrence of some specific event. For example that a specific call has been accepted terminated or non-answered. This means that the Service Switching part is previously instructed by the SCP to perform monitoring for detection of some specific SIP messages. (Ex.: ACK received for a particular *Call-Id*).

When event detection occurs, the Service Switching part notifies the SCP by sending a REGISTER request with the corresponding SIPext header fields.

The REGISTER request sent for event notification contains the following headers (see Fig.6.7):

- *To:*, *From:*, *Call-Id:*, *Cseq:*, carrying call context content. In most cases the content of these headers is not used for the execution of the requested operation, because generally all the call context data needed are present in the *Operation:* header.
- *Oseq:* header used for sequencing the SIPext REGISTER requests and matching the responses.
- *Via:* header specifying that the request has been generated by the Service Switching server. For REGISTER requests that are result of event detection, this is the only one present *Via:* header.
- *Operation:* header, specifying the operation to be executed and its parameters.

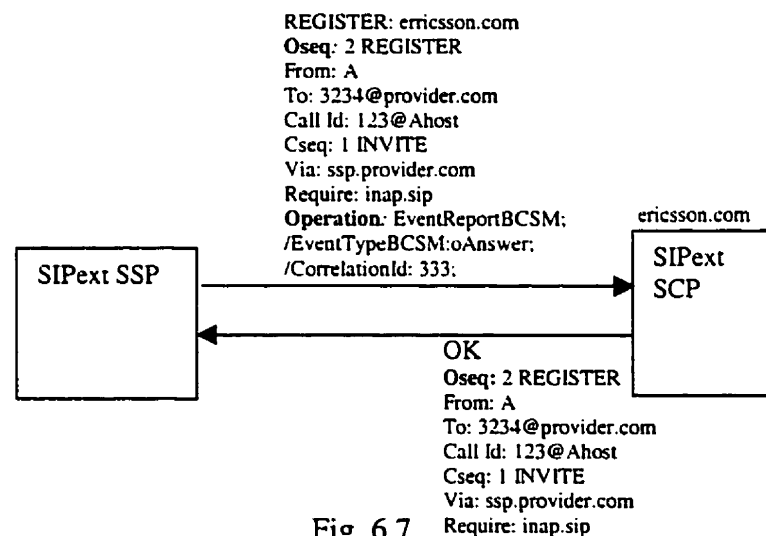


Fig. 6.7

SIPext SSP notifies the SCP about event detection

After sending a REGISTER message requesting an operation to be executed by the SCP, the Service Switching part waits for response message. The response message reports either that:

A. The requested operation has succeeded.

In this case (Fig.6.2) an OK response message is returned. In addition to the pure SIP headers (*To:*, *From:*, *Call-Id*, *Cseq*, *Via:*) and the SIPext *Oseq:* header the OK message can contain also *Operation:* SIPext headers.

If an *Operation:* header is present, then the Service Switching part performs the required actions. For example if the *Operation:* header specifies that the call has to be routed to a forward address, then the Service Switching part formulates the corresponding SIP request and transfers the control to the pure SIP server.

B. The requested operation has not succeeded.

In this case a SIP 400 or 500 response message is returned (Fig. 6.8). The message contains an *Error-Op:* SIPext header specifying the error type and parameters (if any).

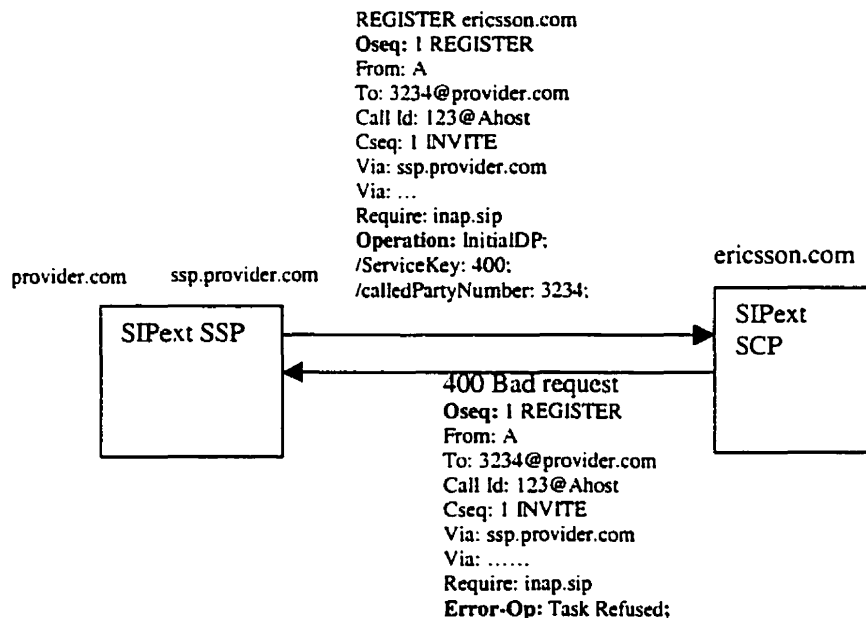


Fig. 6.8

Unsuccessful operation execution

Depending on the *Error-Op:* header and parameters the Service Switching part can perform different actions. For example, it can:

- retransmit the request immediately or after the specified in the *Error-Op:* header time.
- formulate more specific SIP response and transfer it to the pure SIP server for sending it back to the caller.
- etc.

6.2.5.2 SIPext Service Switching part acting as a server

As noted in the description of the behavior of the SIPext SCP, it is possible that it sends to the SIPext Service Switching server a REGISTER message requesting call monitoring and event reporting operation to be performed. In this case the Switching server

- stores the call context data needed for event detection;
- starts monitoring for the specified events;
- formulates an OK response for the REGISTER request.

Commonly the OK response in this case will carry only SIP headers formulated according to the pure SIP rules.

6.3 Examples of IN VAS implemented in Internet Telephony with SIPext protocol.

The Freephone and Call Distribution services will be presented to illustrate how the SIPext tools can be used for Internet Telephony implementation of IN Value Added Services in the hybrid SIP-IN/WIN network scenario.

The VAS implementation in the hybrid SIP-IN scenario reuses the Service Logic Programs of the IN based SCP. This implies that the SIPext messages must correspond to the SSP-SCP INAP messages in the pure IN scenario. Therefore before going into the description of the SIPext VAS implementation we need to present their INAP based implementation.

6.3.1 Freephone Service

Freephone service implemented using INAP

Depending on network-specific mechanisms for charging, there are several possible implementations of the Freephone service in an Intelligent Network environment. For the purpose of this example we will consider the Freephone service in the case when the

charge is computed by the SCP. The following INAP operation sequence describe the example implementation of Freephone service:

Note: → the operation is sent from SSP to the SCP

← the operation is sent from SCP to the SSP

1. → InitialDp(serviceKey = 0800, calledPartyNumber = 3456789)
2. ← Connect(destinationRoutingAddress = 6543210,
correlationId=1111)
RequestReportBCSMEvent(eventTypeBCSM=o_Answer,
monitorMode=notifyAndContinue, eventTypeBCSM=o_Disconnect,
monitorMode=notifyAndContinue, bcsmEventCorrelationId=1111)
3. → EventReportBCSM(eventTypeBCSM=o_Answer, bcsmEventCorrelationId=1111)
4. → EventReportBCSM(eventTypeBCSM=o_Disconnect,
bcsmEventCorrelationId=1111)

Freephone service implemented in SIP based Internet Telephony

User A sitting at host Ahost calls user B with address 8003234@provider.com. The provider server consults a Trigger server and concludes that the user B is subscribed for Freephone service.

The SIPext messages, initially generated as result of encountering a Freephone service trigger, are graphically presented in the Fig. 6. 9.

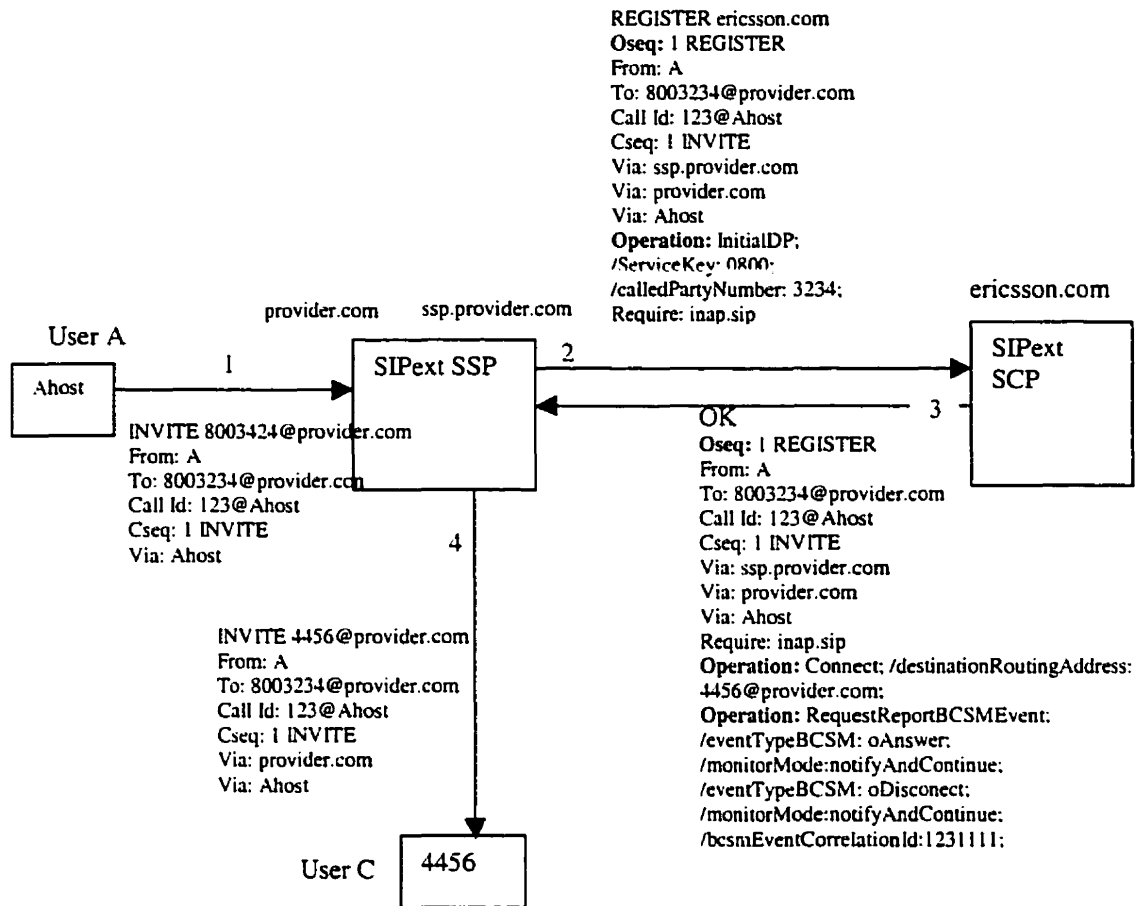


Fig.6.9

User A calls Freephone number 800 3424

The SIPext REGISTER request (message 2) contains two SIPext headers. The *Operation:* header specifies that the Freephone Service Logic Program must be started and carries the call context data needed for its execution. In the example this is the data carried in the Called Party Number parameter. The *Oseq:* header contains the operation sequence number.

On receiving message 2, the SIPext SCP analyzes the SIPext headers and starts the Freephone SLP. After the successful SLP execution, it generates OK response by copying all headers from message 2 except the *Operation:* header. The results of the SLP execution are formulated in two new *Operation:* headers. The first one (Connect

operation) specifies that the call has to be routed to another the destination number. The header Operation: RequestEventReport requests that the SIPext SSP notifies the SCP for an oAnswer (“Call Accepted”) or oDisconnect (“Call Disconnected”) event.

After receiving the OK response, the SIPext SSP executes the operations specified. The Connect operation results in generating a pure SIP INVITE request and sending it over the SIP network to the forward address 4456@provider.com. The RequestEventReport operation instructs the SIPext SSP to start monitoring the messages with the specific *Call-Id* number and to notify the SCP if ACK or BYE request for this *Call-Id* has been received. The ACK request corresponds to the IN oAnswer (i.e. “Call Accepted”) event, and BYE corresponds to the oDisconnect (i.e. “Call Disconnected”) event.

The next Fig. 6.10 describes the SIP messages exchanged in case user C sitting at host 4456@provider.com accepts the forwarded call. At this point of the scenario the SIPext SSP operates as a pure SIP proxy server.

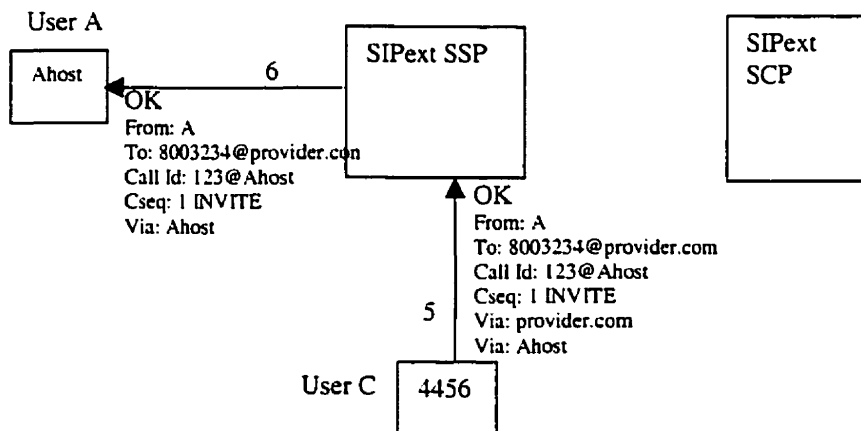


Fig. 6.10

User C accepts the Frephone call

When the SIPext SSP receives the ACK request (message 7 on Fig. 6.11) from the caller’s user agent it performs two actions:

- Proxies the ACK request to user C (message 8).

- Generates a SIPext REGISTER request for and sends it to the SIPext SCP (message 8'). By including BCSMEventReport operation in the REGISTER message, it notifies the SCP that the event oAnswer has occurred for the monitored call.

The SIPext SCP confirms the REGISTER request by responding with OK message (message 9) and starts a SLP that bills the call on the 4456 account.

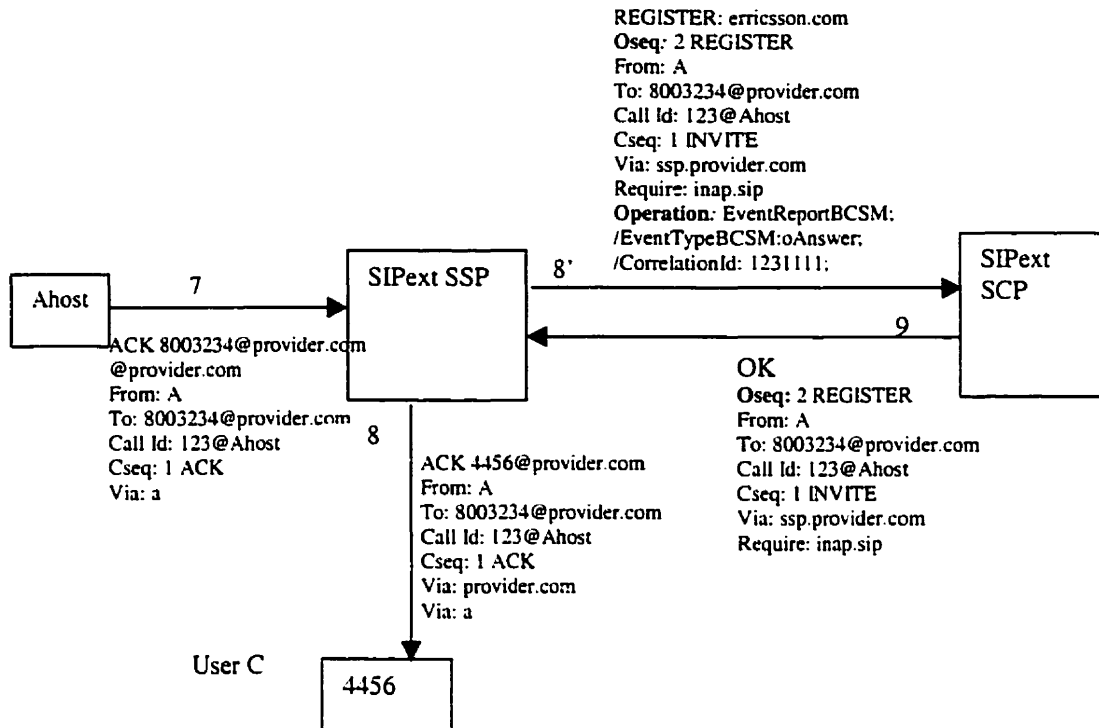


Fig. 6.11

The Freephone call is setup

The SIPext SSP continues monitoring the signaling messages for the *Call-Id:123@Ahost*. At some point of the call, one of the call parties decides to terminate the connection. His user agent issues a BYE request, which corresponds to an oDisconnect event.

The SIPext SSP detects an oDisconnect event either when it proxies a BYE request, or when it proxies an OK response to a BYE request for the monitored *Call-Id*.

Fig. 6.12 presents the case when the caller A issues a BYE request, proxied by the SIPext SSP to the callee C (message 11'). When the BYE request is detected, the SIPext SSP sends to the SIPext SCP a REGISTER request containing ReportBCSMEvent operation

header (message 11). The SIPext SCP confirms the request by sending an OK response and notifies the SLP about the oDisconnect event. The SLP completes the call billing.

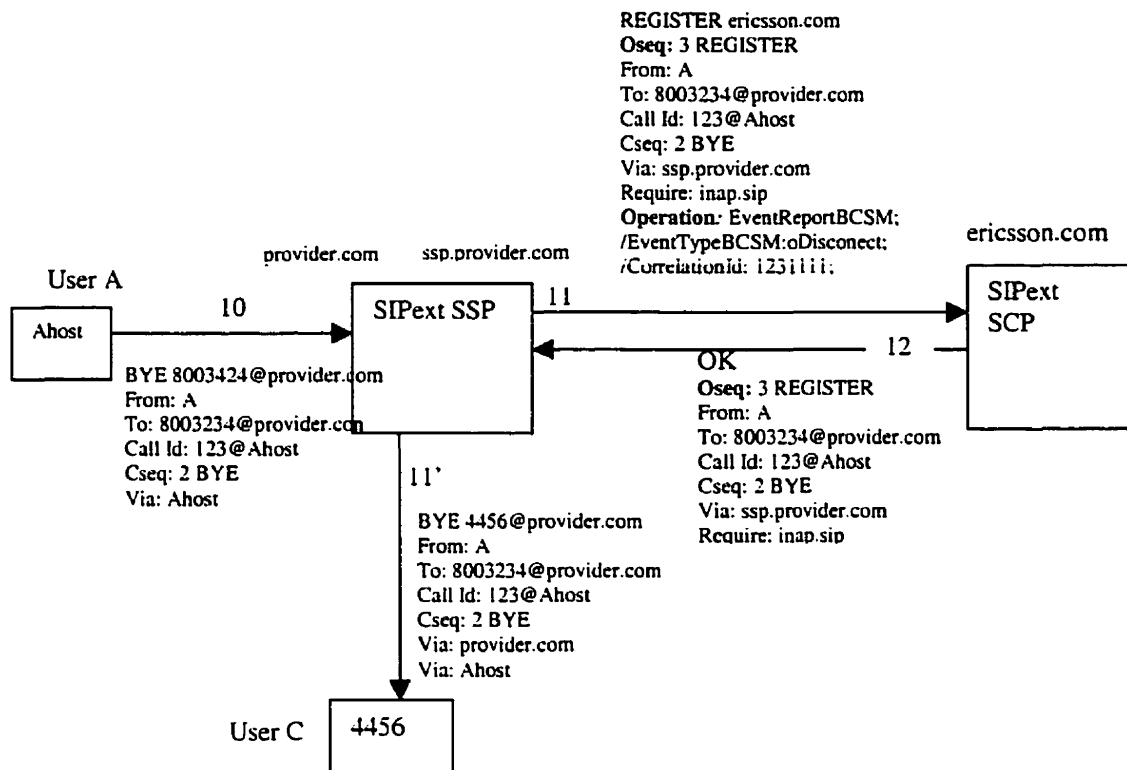


Fig. 6.12

User A terminates the Freephone call

6.3.2 Call Distribution Service

As mentioned in section 1.3.1 the Call Distribution service routes the incoming calls to different destinations depending on user defined rules.

For the purpose of this example, let us assume that user B is subscribed for Call Distribution service and has defined the following rules:

- A. The maximum calls that the number B is allowed to answer are 50 per day.
- B. If the number of calls is more than 50 then the calls are forwarded to number 6543210.

Call Distribution implemented using INAP

A. INAP operation messages exchanged between an SSP and SCP in case the number of connected calls to the number 3456789 has not reached the threshold 50.

1. → InitialDp(serviceKey = 0100, calledPartyNumber = 3456789)
2. ← Connect(destinationRoutingAddress = 3456789, correlationId=1112)
RequestReportBCSMEvent(eventTypeBCSM=o_Answer,
monitorMode=notifyAndContinue, eventTypeBCSM=o_Busy,
monitorMode=notifyAndContinue, eventTypeBCSM=o_NoAnswer
bcsmEventCorrelationId=1112)
3. → EventReportBCSM(eventTypeBCSM=o_Answer, bcsmEventCorrelationId=1112)
or
EventReportBCSM(eventTypeBCSM=o_NoAnswer, bcsmEventCorrelationId=1112)
or
EventReportBCSM(eventTypeBCSM=o_Busy, bcsmEventCorrelationId=1112)

Note that the SCP requests monitoring for three events: oAnswer, oNoAnswer, and oBusy. This is needed, because according the semantics of the Call Distribution service described here, the threshold of 50 applies to calls answered by the subscriber 3456789. Therefore the service related data (i.e. the number of the answered calls) are updated only if an oAnswer event has been reported. On detection of a noAnswer or oBusy event the transaction started with the RequestReportBCSMEvent operation is ended without updating the number of answered calls.

B. INAP operation messages exchanged between an SSP and SCP in case the subscriber 3456789 has already answered 50 calls.

In this case the call is forwarded to the number 6543210.

1. → InitialDp(serviceKey = 0100, calledPartyNumber = 3456789)
2. ← Connect(destinationRoutingAddress = 6543210, correlationId=1112)

Call Distribution service implemented in SIP based Internet Telephony

User A places a call to user B's number 3456789@provider.com, which results in sending an INVITE request (Fig. 6.13). After receiving the INVITE the SIPext SSP server "provider.com" consults a Trigger server and detects that the user 3456789 is subscribed for Call Distribution service. Then the SIPext SSP generates REGISTER request to the SIPext SCP. The *Operation:* header field in the request specifies that the SCP must execute the Call Distribution Service Logic Program.

The Service Logic Program consults the user data and determines whether the connections to the number 3456789 have reached the threshold of 50 or not.

A. The user 3456789 has answered less than 50 calls.

This case is graphically presented on the figures Fig. 6.13 – Fig. 6.16.

The SIPext SCP routes the call to the 3456789 number (See Fig. 6.13) by adding a Connect operation header field in the OK response message. A RequestReportBCSMEvent operation header is also added.

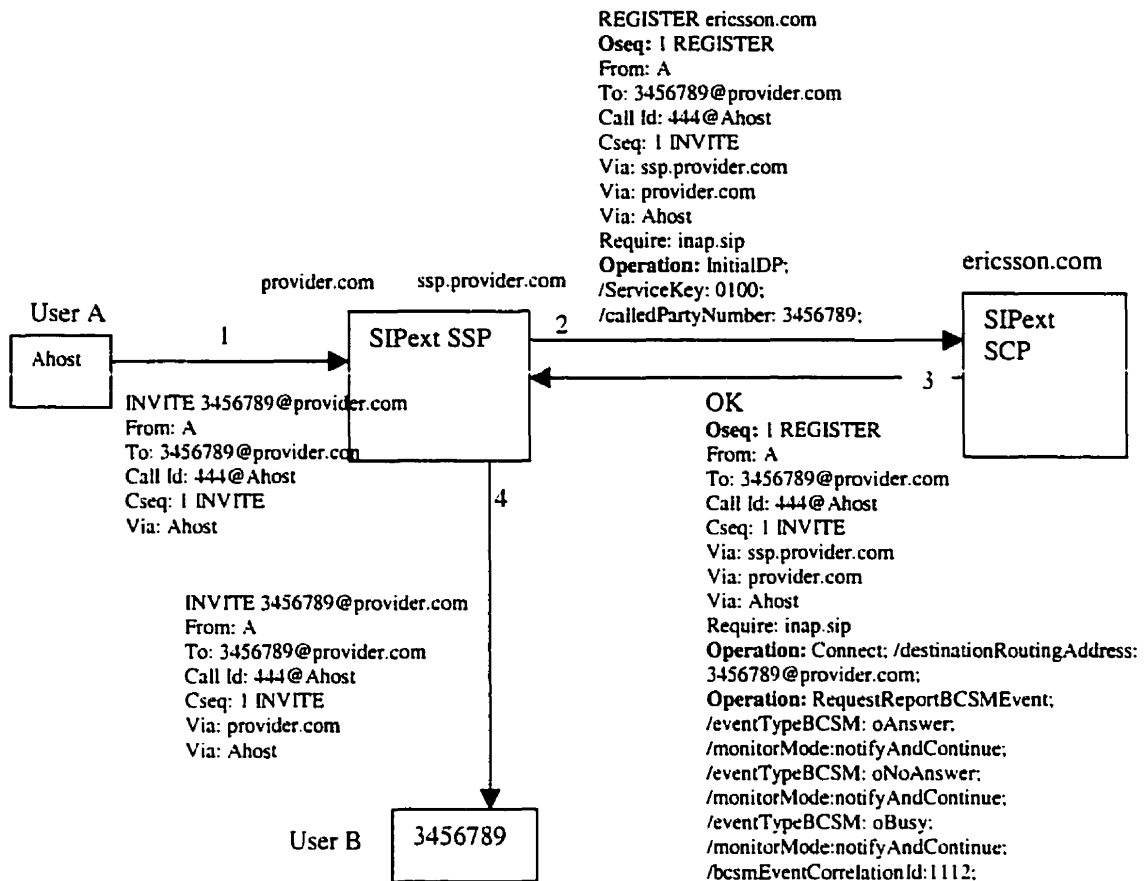


Fig. 6.13

A call attempt to user B, who has answered less than 50 calls for the day

At this point the SIPEXT SSP starts monitoring the *Call-Id*: 444@Ahost.

If it detects an OK response (i.e. oAnswer event) for this *Call-Id*, it generates a ReportBCSMEvent REGISTER message to the SIPEXT SCP (Fig. 6.14).

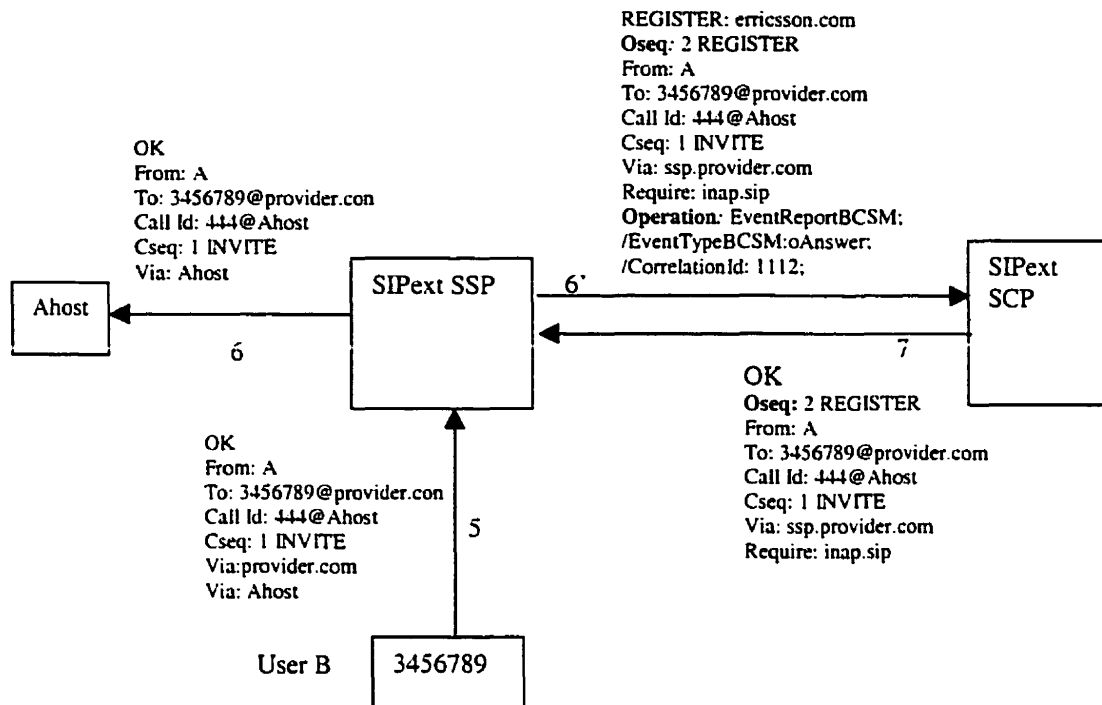


Fig. 6.14

The SIPext SSP detects and reports that user B has answered a call

The behavior of the SIPext SSP is similar (Fig. 6.15), if it detects a busy response (i.e. oBusy event).

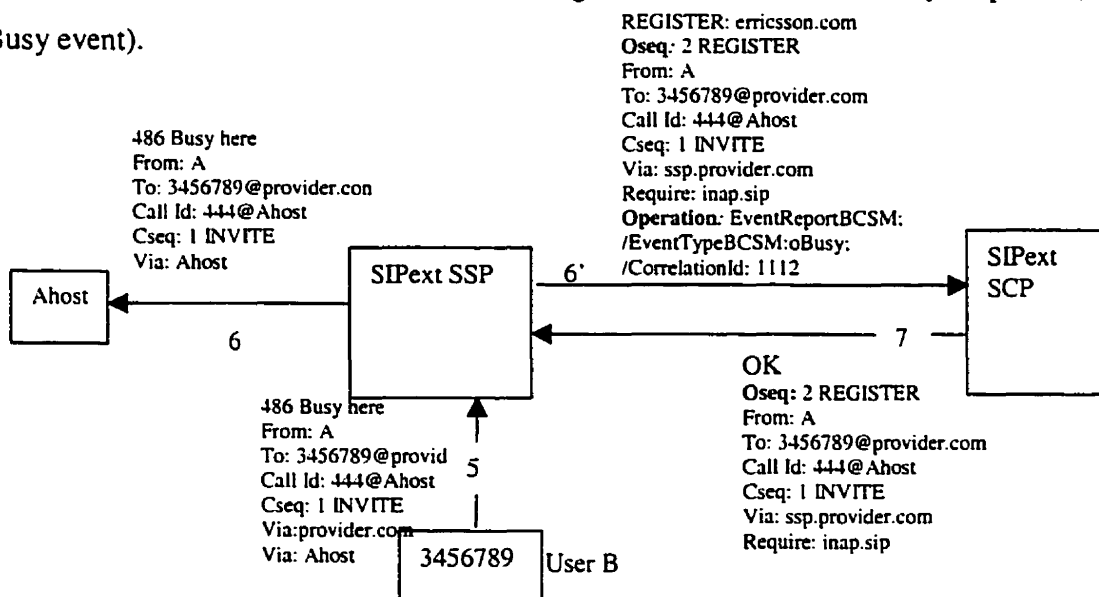
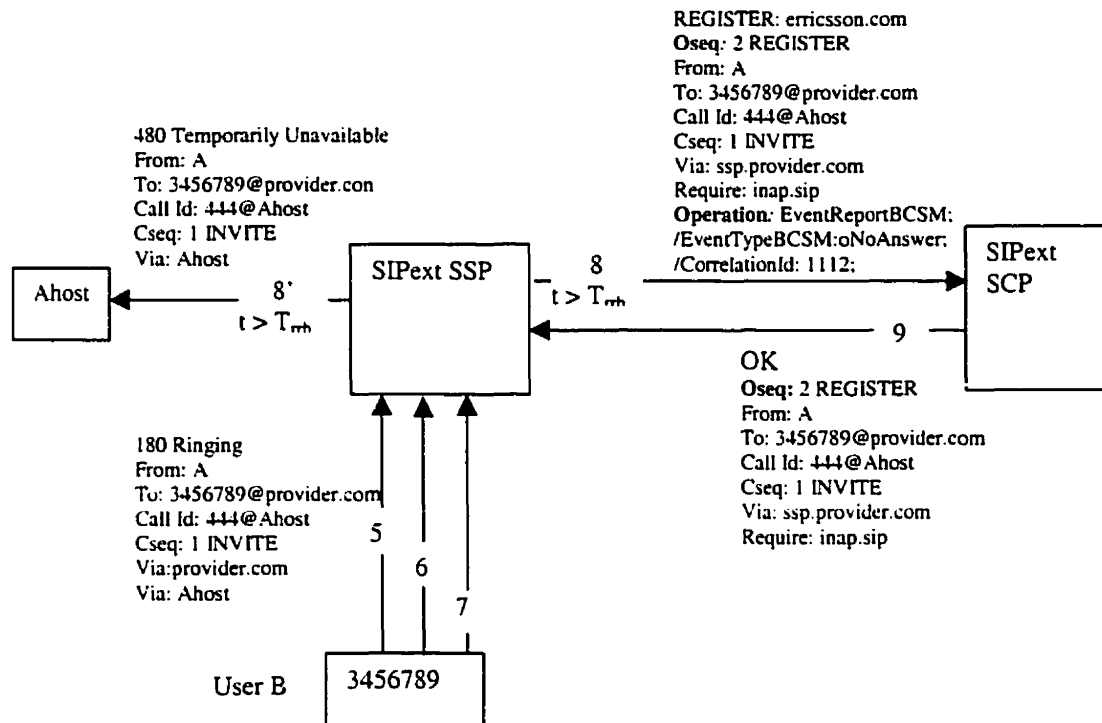


Fig. 6.15

The SIPext SSP detects and reports that user B is busy

The oNoAnswer event is detected when either a 480 Temporarily Unavailable response is received or when an operation related timer $T_{\pi b}$ expires before an OK response has been received.

For example, the SIPext server is receiving informational responses “180 Ringing” from the user B’s user agent (messages 5, 6,7 on Fig. 6.16).



User B does not answer

In the example it is supposed that the message 7 is the last one received before the timer $T_{\pi b}$ expires. After $t > T_{\pi b}$ the SIPext SSP reports an oNoAnswer event to the SCP and sends a 480 Temporarily Unavailable response to the user A.

B. User B has already answered 50 calls.

In this case the SIPext SCP instructs the SIPext SSP to route the call to the forward number 6543210 (See Fig. 6.17) by including Connect operation header in the OK REGISTER response. From this point on the SIPext SSP handles the call as a pure SIP server.

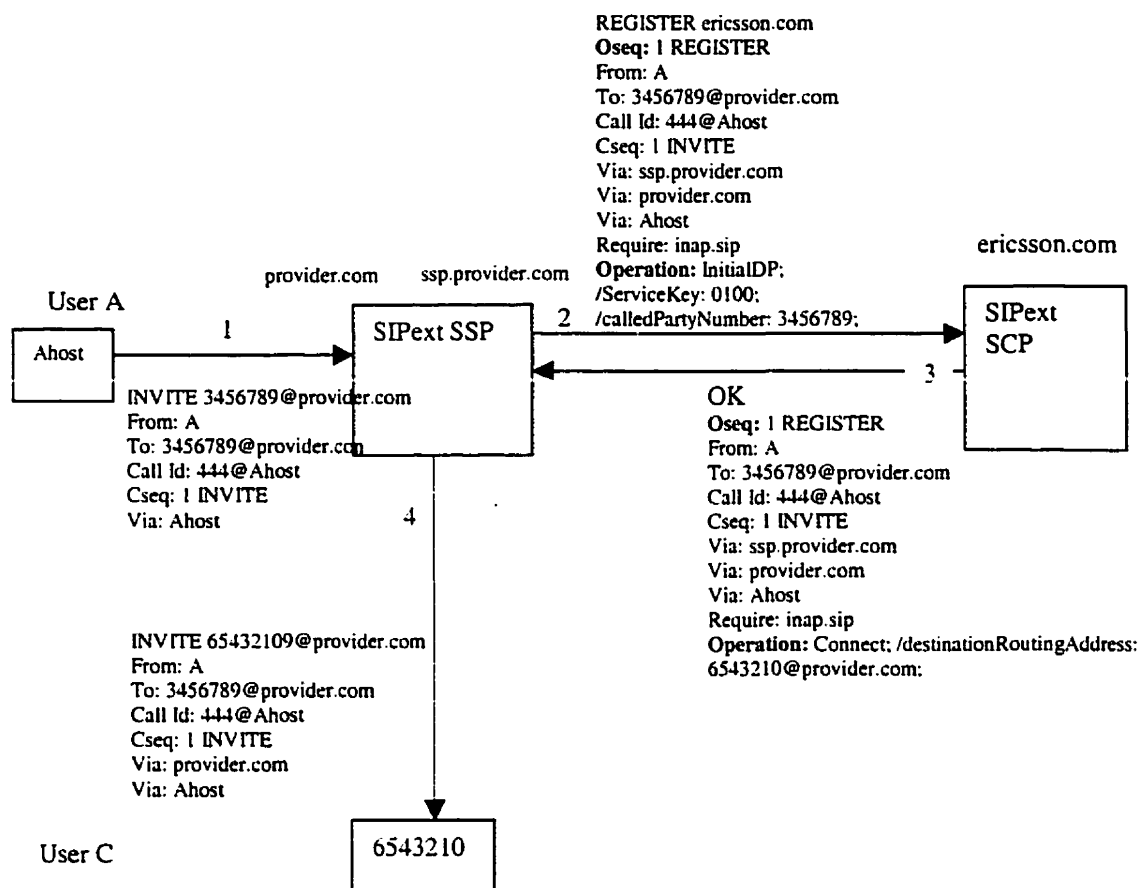


Fig.6.17

A call attempt to user B, who has already answered 50 calls for the day

6.4 Conclusion

Defining a SIP extension is one of the possible ways to provide for SIP network access to the IN VAS. A strong argument for the development of SIP based interface is that SIP is the protocol inherent to one of the communicating entities – the SIP network server. When implementing this interface we also benefit from the fact that SIP is already a built protocol, which offers rich extensibility mechanisms. The Freephone and Call Distribution service examples show how, by using the defined SIP extension inap.sip, we are able to provide for the communication between SIP network servers and IN Service Control Point and to bring the IN VAS to the SIP network users.

Chapter 7

Implementation issues

The previous chapter proposed an extension to the SIP protocol in terms of extension headers, network entities and basic operation. We can generalize the scope of the work presented above as a design solution for the communication interface between a SIP proxy server and IN SCP from architectural and operational perspective. Naturally the next phase in the investigation of the hybrid SIP-IN VAS approach will be the implementation of the proposed SIPext solution. This chapter will stress some issues related to this implementation.

As described in Chapter 6, the SIPext communication interface between the SIP proxy and the IN SCP is supported by the SIPext Service Switching part of the, so called, SIPext SSP and the SIPext part of the SCP. The diagram on Fig. 7.1 presents the set of functional modules that build the SIPext parts of the SSP (in our case this is the SIP proxy) and the SCP.

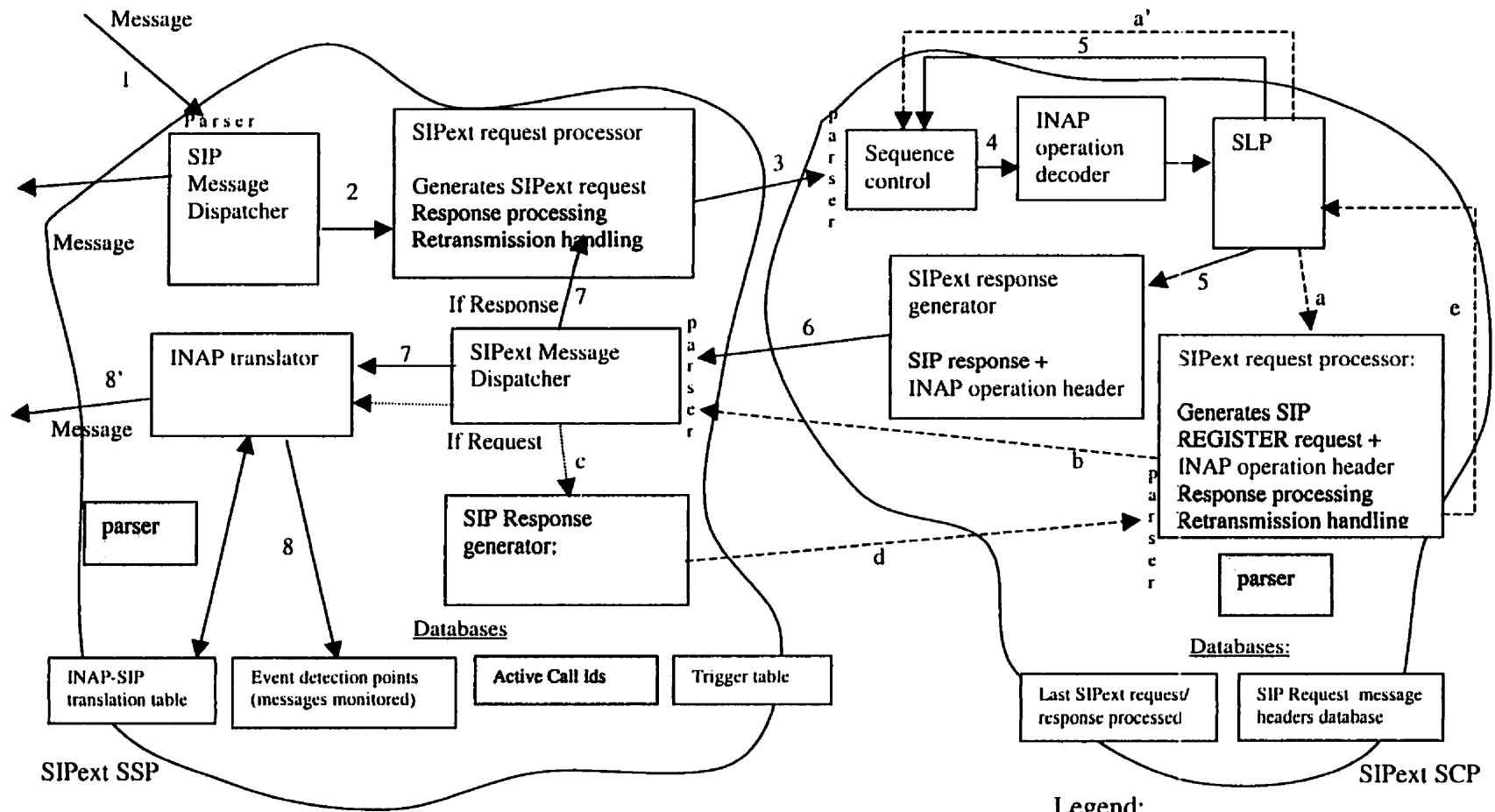


Fig. 7.1

Functional modules of the SIPext SSP and SIPext SCP

Their functions of the SIPext SSP and SIPext SCP modules and the databases supported or needed for their operation are described in more detail below.

SIPext SSP

SIP Message Dispatcher

Functions:

- service trigger detection

If the message is an INVITE and has new Call Id, then it is checked against the service trigger table.

- event detection

If the message is not a service trigger, then it is checked against a list of expected messages (events).

If it is not an armed (expected) event either then it is proxied over the SIP network

Databases accessed:

- SIP server database of active Call Ids.
- Service triggers table
- Database of the armed events (messages) and their arming (the corresponding INAP operations to be executed).

SIPext request (INAP operation) processor

Functions:

- Formulates the Operation header field and generates a SIPext REGISTER request as result of detection of an event or trigger by the SIP Message Dispatcher.
- Processes the response to the request as defined in SIP, i.e matches responses to requests, handles request retransmission, etc.

SIPext Message Dispatcher

This is the entry point of the SIPext messages sent by the SCP.

Functions:

- Classifies the messages as Response or Request.

- If Response, notifies the SIPext request processor and transfer the message data to the INAP-SIP translator.
- If Request, the SIP headers are transferred to the SIP response generator. The SIPext message data are transferred to the INAP-SIP translator.

INAP translator

Functions:

- Translates the INAP operation and its parameters either to a SIP message or to an instruction for updating the Events detection points database.

Database accessed:

- INAP-SIP translation table

This table contains the knowledge needed to translate an INAP operation to the corresponding action of the SIPext SSP.

SIPext SCP

Sequence Control

Function:

- Maintains the database storing the last SIPext request and response processed.
- Ensures that the INAP operations are transferred to the SLP in the order that they were issued.

INAP operation decoder

Functions:

- Decodes the INAP operation header.
- Formulates the INAP operation according the SLP interface.

SIPext response generator

Functions:

- Formulates the result of the SLP execution as an SIPext header field.
- Consults the requests database.
- Builds an SIPext response message and sends it to the SSP.

Databases accessed:

- SIP Request message headers database.

This database keeps the SIP related headers for each request that is currently processed.

SIPext request processor

Functions:

- Formulates the INAP operation header, when the SLP requires an INAP operation to be executed by the SIPext SSP.
- Generates the SIPext request (SIP REGISTER request + *Operation:* header) and sends it to the SSP (SIPext Message Dispatcher).
- Processes the response to the request as defined in SIP, i.e. matches responses to requests, handles request retransmission, etc.

SLP

The Service Logic Programs represent the IN based implementation of the services residing on a SCP. For the purpose of prototype building and testing the SLP module may represent a simulation of the IN based SLPs.

The data carried in the Operation header field specify which SLP has to be executed. After the SLP execution the SIP request/response databases are updated and the execution results are transferred to the SIPext response generator or to the SIPext request processor.

Databases accessed:

Service data residing either on IN Service Data Point or in database simulating it (not shown on the Fig.7.1).

Here are some of the issues that have to be considered in the process of the implementation design of the SIPext entities:

- The SIPext SSP and the SIPext SCP are located on different network entities. The SIPext SSP can be seen as an upgrade to a SIP proxy server, while the SIPext SCP as upgrade of an IN SCP. Nevertheless for prototyping and testing it is feasible that we start by implementing the SIPext SSP and the SIPext SCP on the same physical entity, which naturally can be the SIP proxy. For this purpose we will need to simulate the operation of the Service Logic Programs, which in the real implementation are the already implemented and reside on the IN SCP.
- As SIPext is built on top of SIP, the SIPext SSP and the SIPext SCP will include functional modules supporting the SIP operation.
Therefore it is reasonable that in their implementation we reuse some modules of the proxy implementation. For example these may be the shaded blocks on Fig.7.1.
- A substantial part of the implementation will represent a design of the INAP-SIP translation table, Event detection point database and Trigger table.
The Event detection point database can be designed as table containing the messages that has to be detected. These messages are identified by a list of key headers and the expected values of their parameters. For each message monitored the corresponding INAP operation header is specified as well as the SIP header's parameters that have to be passed as INAP operation parameters.
The Trigger table can be located either on a remote network node (as in the network scenario illustrated on Fig. 5.1), or it can be collocated with the SIPext SSP (as presented on Fig.7.1). Generally the Trigger table can follow the model of the Event detection points table. There are two differences between them, which although not substantial, have to be mentioned:
 - In the Trigger table the only messages listed for monitoring are INVITE requests, while in the Event table the specified messages can be any SIP request or response.
 - The Trigger table is static in the sense that it is created and updated only by the service provider. The Event detection points table is dynamic, i.e. it is updated in the course of the Service Logic Program execution.

Finally we want to stress again that the starting point for any implementation for the purpose of testing or prototyping of the SIP inap.sip extension or the hybrid SIP-IN architecture is the existing SIP proxy server implementation, providing the SIPext SSP with the SIP functionality.

Chapter 8

Conclusion and future work

Bringing the Internet technology to the Telecommunications world is a challenging but worthy task. It involves building a new protocol and network architecture that is able to provide for the same level of quality, reliability and services sophistication as the Classical Telephony offers today. In return an IP based telephony is promising low cost connections, easy service deployment, end user service control, natural computer-telephony and data/voice integration. But there are many avenues to be explored before getting to an IP based telephony network able to compete with the existing classical telephony and today we witness a lot of efforts devoted to research in this area.

This thesis aimed to be a small contribution to these research efforts focusing on the Internet Telephony Value Added Services. The first chapters give the framework for the further study by building a background on the Intelligent Networks in the Classical Telephony and the Internet Telephony. After briefly presenting the two alternative signaling protocols H.323 and SIP, the study takes the path of SIP, SIP based Internet Telephony and Value Added Services. An overview and analysis of three possible SIP network scenarios for service provisioning is presented. Next the study focuses on the service implementation in a SIP network and specifically on two of the possible approaches to the service implementation. Then a conceptually different approach to providing services in the Internet Telephony, called the hybrid SIP-IN approach, is described and is chosen as a subject for the further investigation. It is based on reusing the Service Logic Programs stored on the Intelligent Network Service Control Points (SCPs) and has to be studied in several aspects. The last chapters are devoted to one of them - the communication interface between the SCP and the SIP server. They present an architecture design of a communication protocol built as a SIP extension and address some implementation issues relating to it. There is much more, left out of the scope of this work, that is part of the study of the presented hybrid SIP-IN approach to VAS.

Based on the designed protocol architecture presented here, the study of the hybrid SIP-IN approach to VAS implementation may be continued in the following directions:

- Upgrade of a SIP proxy server with a SIPext module, interfacing the SIP proxy server to the the IN SCP.
- Implementation of a SIPext module interfacing the Service Logic programs to the SIP network.
- Implementation of a modules simulating the Service Logic Programs of a SCP.
- Implementation of a module simulating a Trigger server.
- Analysis of the complexity of the implementation
- Testing of the hybrid VAS implementation
- Analysis of the designed protocol
- Assessing the feasibility of the approach
- If the approach is feasible to be implemented, upgrade of an IN SCP with the SIPext module, allowing the SIP network access to the Service Logic Programs.

The steps mentioned above can be detailed more and sub-steps can be specified. The list can also be continued further, but it would go very far from the point, which was defined as a scope of this thesis.

As it was pointed in the beginning, nobody can tell whether, how or what could impose the Internet Telephony as the predominant technology in the Telecommunications world. But it is up to us to look for and explore the alternatives.

Bibliography

- [BER98] P. Bernier The standards struggle, May 1998,
<http://www.soundingboardmag.com/articles/851feat2.html> .
- [BRA98] J. Brancheau, "H.323 Protocol Stack Tutorial", 1998,
<http://www.colorado.edu/infos/jcb/sinewave/network/h323/index.html> .
- [CCS98] H. Schulzrinne, J. Rosenberg, "SIP Call Control Services", Internet draft, IETF, MMUSIC Working Group, March 1998, work in progress.
- [CGIS98] J. Lennox, J. Rosenberg, H. Schulzrinne, "Common Gateway Interface for SIP," Internet Draft, Internet Engineering Task Force, Nov. 1998. Work in progress.
- [CPL99] J. Lennox, H. Schulzrinne, "CPL: A language for user control of Internet telephony services", Internet draft, IETF, IPTEL Working Group, February 1999, work in progress.
- [CPLR98] J. Lennox, H. Schulzrinne, "Call Processing Language requirements", Internet draft, IETF, IPTEL Working Group, July 1998, work in progress.
- [CSH98] H. Schulzrinne and J. Rosenberg, "A Comparison of SIP and H.323 for Internet Telephony," in Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), (Cambridge, England), July 1998.
- [DCWP98] Databeam Corporation White Paper, May 1998,
<http://www.databeam.com/h323/h323primer.html> .
- [FAR] T. Farley, "Telephone History",
<http://www.privateline.com/TelephoneHistory2/History2.html> .
- [FAY97] I. Faynberg, L. Gabuzda, "The IN standards. Their application to services", McGraw Hill, 1997, ISBN 0-07-021422-0.
- [GLI99] R. Glitho, "Advanced Services Architectures for Internet telephony: State of the Art and Prospects", 1999, (submitted for eventual publication in IEEE Network special issue on Internet telephony).
- [GTH96] G. Thom, H.323: The Multimedia Communications Standard for Local Area Networks, IEEE Communications Magazine, pp. 52 – 56, December 1996.

[INT99] IN tutorial, International Engineering Consortium, 1999, <http://www.webproforum.com/bellcore2> .

[IP/IN99] Internet Protocol/Intelligent Network Integration Tutorial, International Engineering Consortium, 1999, <http://www.webproforum.com/microlegend/topic01.html>.

[ITAP98] H. Schulzrinne, J. Rosenberg, "Internet Telephony: Architecture and Protocols - an IETF perspective," Computer Networks and ISDN Systems, vol. 31, pp. 237-255, February 1999.

[ITRT98] P. Friedman, M. Sargent, "How will Internet Telephony Revolutionize Telecommunications?", Internet Telephony Round Table, First quarter 1998, <http://www.tmcnet.com/articles/itmag/FirstQ/q1roundtable.htm> .

[ITT99] Internet Telephony Tutorial, International Engineering Consortium, 1999, <http://www.webproforum.com/siemens2/> .

[LSL99] J. Lennox, H. Schulzrinne, and T. F. L. Porta, "Implementing Intelligent Network Services with the Session Initiation Protocol," Technical Report CU-CS-002-99, Columbia University, New York, New York, January 1999.

[MAG96] T. Magedanz, R. Popescu, "Intelligent Networks – Basic Technology, Standards and Evolution", 1996, International Thomson Computer Press, ISBN 1-85032-293-7.

[MOD90] A. Modarressi, R. Skoog, "Signalling System No 7: A Tutorial", IEEE Communications Magazine, pp.19-35, July 1990.

[NEU92] G. Neufeld, S. Vuong, "An Overview of ASN.1", Computer Networks and ISDN Systems, vol. 23, pp.393-414, 1992.

[Q.1208] ITU-T Recommendation Q.1208: General Aspects of INAP, 1993.

[Q.1218] ITU-T Recommendation Q.1218: Interface Recommendation for IN CS-1, 1996.

[Q.1219] ITU-T Recommendation Q.1219: Intelligent Network user's guide for CS-1, 1994.

[RAJ98] G. S. Raj, "A detailed comparison of Corba, DCOM and Java/RMI", September 1998, <http://www.execpc.com/~gopalan/misc/compare.html> .

- [rfc1945] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol - HTTP/1.0", Network Working Group, May 1996, work in progress,
<http://www.informatik.uni-halle.de/www/http/draft-ietf-http-spec.html> .
- [rfc2543] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, "SIP: Session Initiation protocol", Network Working Group, March 1999.
- [RSUA98] H. Schulzrinne, J. Rosenberg, "Requirements for SIP servers and User agents", November 1998.
- [SATS98] H. Schulzrinne and J. Rosenberg, "The Session Initiation Protocol: Providing Advanced Telephony Services Across the Internet," Bell Labs Technical Journal, vol. 3, pp. 144-160, October-December 1998.
- [SIT98] H. Schulzrinne, J. Rosenberg, "Signaling for Internet Telephony," International Conference on Network Protocols (ICNP), (Austin, Texas), October 1998.
- [THO94] J. Thormer, "Intelligent Networks", 1994, Artech House, Inc., ISBN 0-89006-706-6.
- [TUCI99] J. Lennox, H. Schulzrinne, "Transporting User Control Information in SIP REGISTER payloads", Internet draft, IETF, IPTEL WG, February 1999, work in progress.
- [X.690] ITU-T Recommendation X.690: Information Technology - ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), 1997.
- [X.691] ITU-T Recommendation X.691: Information technology - ASN.1 encoding rules - Specification of Packed Encoding Rules (PER), 1997.