# Framework for Development of Advanced Telecommunication Services in Current and Future Converged Networks

**Soler, José; Dittmann, Lars; Fosgerau, Anders**

*Publication date:*
2006

*Document Version*
Publisher's PDF, also known as Version of record

# Framework for Deployment of Advanced Telecommunication Services in Current and Future Converged Networks.

José Soler


Research Center COM
Technical University of Denmark
April 2005

Supervisors:
Lars Dittmann
Anders Fosgerau

# Abstract

This thesis presents different experiences related to architectures and mechanisms for deployment of telephony services, understood as especial features complementing the basic voice service. The context for these experiences is the transition of telecommunication (telephony) networks from circuit switched based systems towards packet based ones.

Service deployment in a specific hybrid PSTN/IN/VoIP architecture is presented as well as a description of the enabling technologies. Discussion on service implementation examples is provided.

The convenience of network neutral service invocation is introduced and how this has been achieved, by means of Web Services-based mechanisms. A single-request / single response protocol for invocation of a specific set of services is presented and a decoupling mechanism between signalling network protocols and service logic invocation demonstrated. Services, have been designed, implemented and tested as part of this demonstration. Performance characterization of different service implementations is also provided.

Finally, a data model for an overlay network to existing telecommunication (telephony) infrastructures, allowing operational convergence is described.

# Resumé

Denne afhandling præsenterer forskellige erfaringer relateret til arkitekturer og mekanismer til udrulning af telekommunikationstjenester, her forstået som tjenester udover basal talekommunikation. Dette skal ses i lyset af telekommunikationens (telefoniens) udvikling fra kredsløbskobling til pakkekobling.

Udrulning af tjenester i specifikke, hybride PSTN/IN/VoIP arkitekturer præsenteres sammen med en beskrivelse af de mulighedsskabende teknologier. En diskussion af tjenesteimplementering er inkluderet.

Fordelene ved netværksneutral tjenesteudrulning introduceres sammen med en beskrivelse af hvordan dette er opnået ved at benytte web baserede tjenester. En single-request / single-response protokol beregnet til invocation af en specifik gruppe af tjenester præsenteres og der demonstreres en mekanisme, der kan gøre netværkssignalering uafhængig af service-logic-invocation. Som del af denne demonstration er tjenester blevet designet, implementeret og afprøvet. Der fremlægges performancekarakteristikker af forskellige tjenesteimplementeringer.

Slutteligt findes en beskrivelse af en datamodel til et overlay-netværk til eksisterende telekommunikationsinfrastrukturer, som vha. datacentralisering giver operationel konvergens.

# Acknowledgements

Firstly I would like to thank Lars Dittmann for giving me the chance to work in the Networks area at COM. Without any doubt his gesture, trusting a crazy guy writing him from the Far East, has changed my life more than what I ever imagined and I will be able to thank.

I owe sincere gratitude to Anders Fosgerau. Not only he managed to find me a dream place to live in, when I arrived to Denmark, but during the three years we shared the office he provided me with daily candies, cookies and a never ending positive attitude, while standing my *plinki-plinki* music and my not always stable *spansk* temper. As a direct co-supervisor, his outstanding technical knowledge and capabilities turn his advices into the foundation for my confidence in the work I had to undertake.

My gratitude also, to the rest of members of the Networks area, for making the shared, working and non-working moments an enjoyable experience, *selv om hvis det var på dansk*. Henning Christiansen must be acknowledged for his excellent mood whenever he was requested to provide IT support.

All the people involved in the IST GEMINI and FlexiNET projects deserve to be acknowledged for the valuable collaboration that made possible the experiences described in the thesis.

My gratitude to Boris Grabner, Henrik Christiansen and Michael Berger who proofread this document, providing excellent comments and advice to improve the earlier versions of this thesis.

Katja and Jan were my family the two years we spent living together. Without Katja's cares and meals and Jan's gin-tonics, the experience would probably have been not that healthier and definitely much more sad.

Jorge must be specially acknowledged. Without his friendship, good humour and support it would have been difficult to bear the darkness of the mood and winters. Also for having rescued me away from my computer one Friday evening no one expected to be so fruitful. To him and Camilla, *always, always, always* my gratitude.

The Kouassi-Zessia family has taken me in, as an *unzungu* son and brother. Their love and support is herein acknowledged.

My parents and sister deserve a special mention for their unconditional love and constant care through the distance. A number of important things have happened during these years and, despite the troublesome

feelings I am sure they were assaulted by, I always had their advice, backup and support for all my decisions. Furthermore, the timely “ammunition” they have sent in the form of s*errano* ham has made things, if not easier, at least tastier.

Rachel, with her sweet and constant smile, despite my stress and low *cascarrabias* mood, has been taking care of me during the last four months. Without her love, care, confidence and patience this work would have not been possible.

# Table of Contents

# List of Figures

# List of Acronyms

| | |
|---|---|
| **3GPP** | 3rd Generation Partnership Project |
| **ADSL** | Asynchronous Digital Subscriber Line |
| **API** | Applications Programming Interface |
| **ARPA** | Advanced Research Projects Agency |
| **ASN.1** | Abstract Syntax Notation #1 |
| **BCP** | Basic Call Process |
| **BCSM** | Basic Call State Model |
| **BSC** | Base Station Centre |
| **BTS** | Base Transceiver Station |
| **CAMEL** | Customized Applications Mobile Enhanced Logic |
| **CAP** | CAMEL Application Part Protocol |
| **CbCF** | Calendar-based Call Forwarding |
| **CCF** | Call Control Function |
| **CCIS** | Common Channel Interoffice Signaling |
| **CGI** | Common Gateway Interface |
| **CO** | Central Office |
| **CORBA** | Common Object Request Broker Architecture |
| **CS** | Capability Set |
| **DGWN** | Data Gateway Node |
| **DLR** | Destination Local Reference |
| **ENUM** | Telephone Number Mapping |
| **E-SCP** | Enhanced Service Control Point |
| **ETSI** | European Telecommunications Standardisation Institute |
| **FAN** | FlexiNET Access Node |
| **FLAS** | FlexiNET Application Server |
| **FlexiNET** | Flexible Network and Gateways architecture for Enhanced Access Network Services and Applications |
| **FUAN** | FlexiNET UMTS Access Node |
| **FWAN** | FlexiNET WLAN Access Node |
| **GAI** | Generalized Applications Interface |

| | |
|---|---|
| **GDI** | Generalized Data Interface |
| **GEMINI** | Generic Architecture for customized IP-based IN services over hybrid VoIP and SS7 |
| **GGSN** | Gateway GPRS Support Node |
| **GIN** | GEMINI Intelligent Network |
| **GPRS** | General Packet Radio Service |
| **GSL** | Global Service Logic |
| **GSM** | Global System for Mobile Communications |
| **GUP** | Generic User Profile |
| **HFC** | Hybrid Fiber-Coaxial |
| **HLR** | Home Location Register |
| **HOL** | Head of line blocking |
| **HTML** | HyperText Markup Language |
| **HTTP** | HyperText Transfer Protocol |
| **IDE** | Integrated Development Environment |
| **IETF** | Internet Engineering Task Force |
| **IIOP** | Internet Inter-ORB Protocol |
| **IMS** | Internet Multimedia Subsystem |
| **IN** | Intelligent Network |
| **INAP** | Intelligent Network Application Part Protocol |
| **INCM** | Intelligent Network Conceptual Model |
| **IP** | Internet Protocol |
| **IST** | Information Society Technologies |
| **ISUP** | Integrated Services User Part Protocol |
| **ITU** | International Telecommunications Union |
| **J2SE** | Java 2 Platform, Standard Edition |
| **JAIN** | Java Applications for Integrated Networks |
| **JDO** | Java Data Objects |
| **JTAPI** | Java Telephony API |
| **LAN** | Local Area Network |
| **LLC** | Link Layer Control |
| **MAP** | Mobile Application Part Protocol |
| **MCU** | Multipoint Control Unit |
| **MG** | Media Gateway |

| | |
|---|---|
| **MS** | Mobile System |
| **MSC** | Mobile Switching Centre |
| **MTP** | Message Transfer Part Protocol |
| **NSAPI** | Network Service Access Point Identifier |
| **ODL** | Object Description Data Language |
| **ODMG** | Object Data Management Group |
| **OQL** | Object Query Language |
| **ORB** | Object Request Broker |
| **OSA** | Open System Architecture |
| **OSI** | Open Systems Interconnection |
| **PDP** | Packet Data Protocol |
| **PE** | Physical Entity |
| **PINT** | Public Switched Telephone Network Internet Internet-working |
| **PLMN** | Public Land Mobile Network |
| **POI** | Point of Initiation |
| **POR** | Point of Return |
| **PSTN** | Public Switched Telephone Network |
| **PTMSI** | Packet Temporal Mobile Subscriber Identifier |
| **QoS** | Quality of Service |
| **RAF** | Repository Access Function |
| **SAPI** | Service Access Point Identifier |
| **SCCP** | Signalling Connection Control Part Protocol |
| **SCF** | Service Control Function |
| **SCG** | Service Control Gateway |
| **SCP** | Service Control Point |
| **SCTP** | Stream Control Transmission Protocol |
| **SDF** | Service Data Function |
| **SF** | Service Feature |
| **SG** | Service Gateway |
| **SGSN** | Serving GPRS Support Node |
| **SIP** | Session Initiation Protocol |
| **SLR** | Source Local Reference |
| **SM** | Service Management |

| | |
|---|---|
| **SMF** | Service Management Function |
| **SN** | Storage Network |
| **SNMP** | Simple Network Management Protocol |
| **SOAP** | Simple Object Access Protocol |
| **SP** | Signalling Point |
| **SPAN** | Services and Protocols for Advanced Networks |
| **SPIRITS** | Services in the PSTN/IN Requesting Internet Services |
| **SRF** | Specialised Resource Function |
| **SS7** | Signalling System #7 |
| **SSARI** | SIP Servlet API Reference Implementation |
| **SSP** | Service Switching Point |
| **STP** | Signalling Transfer Point |
| **TCAP** | Transaction Capabilities Application Part Protocol |
| **TCK** | Technology Compatibility Kit |
| **TCP** | Transmission Control Protocol |
| **TINA** | Telecommunications Information Networking Architecture |
| **TIPHON** | Telecommunication and Internet Protocol Harmonization Over Networks |
| **TISPAN** | Telecoms and Internet Services and Protocols for Advanced Networks |
| **TUP** | Telephony User Part Protocol |
| **TV** | Tele-Voting |
| **UA** | User Agent |
| **UDDI** | Universal Description, Discovery and Integration |
| **UDP** | User Datagram Protocol |
| **UML** | Unified Modelling Language |
| **UMTS** | Universal Mobile Telephone Service |
| **UTRAN** | UMTS Terrestrial Radio Access Network |
| **VLR** | Visitor Location Register |
| **VoIP** | Voice over Internet Protocol |
| **WLAN** | Wireless Local Area Network |
| **WSDL** | Web Services Description Language |
| **WWW** | World Wide Web |
| **XML** | Extensible Markup Language |

# 1. Introduction

*"¿Cúalo semos? ¿Anóne vinimos? ¿Paqué tanto?*
Alberto Calvo ("Supermaño")

*"Du skal ikke tro, du kan lære os noget"*
Axel Samlemose (Janteloven, "En flygtning krydser sit spor")

Two excerpts from different sources are welcoming the readers of this introduction chapter. The first one, taken from the comic "Supermaño" by Alberto Calvo, can well be used to describe the contents of this thesis. An English translation of the excerpt, though not easy to catch and reflect the slight semantic differences of the on-purpose wrong use of the grammatical elements, could be the following:

*What it are we? Where are we going to, from? Why that much?*

In order to provide a complete answer to the first of the questions in the quote, this introductory chapter tries to give a brief overview on what this thesis is about and where it fits in the current telecommunication trends.

The thesis you are about to read deals with telephony services, understood in the whole of the document as additional features to the basic voice transport in telephony networks. During the rest of the document, different experiences related to telephony services implementation and deployment, are presented and discussed. This provides a description of a likely evolutionary path in the telephony services realm, responding to the second question of Supermaño's quote and to the title of this thesis. The response to the third question of the quote, the reasons behind the choices taken and the conclusions achieved will be hopefully clear to the reader when parsing over the different chapters. Or at least that was the author's intention. And here is where the second quote chosen for the opening of this chapter needs to be explained. Taken from the Danish novel "A refugee looping his track", if I am allowed to translate like that "En flygtning krydser sit spor", by Axel Sandemose where he sets the Janteloven or basic rules characterising the way Danes feel and behave. An approximate translation of the sentence, the $10^{th}$ and last "commandment" of the Janteloven, could be the following:

*Do not think that you can teach us anything.*

This excerpt has been chosen to express the attitude of the author while writing the thesis and during the previous three working years it summarizes. The final aim has not been to outline neither demonstrate a definitive, absolute way of proceeding in order to implement and provide telephony services in current and future networks. On the contrary, the thesis accounts for the problems faced and the solutions adopted during different experiences while working on telephony services related projects. Two european information society technologies (IST) projects, GEMINI and FlexiNET, have provided the ground for these experiences.

For more than a decade, telecommunication operators have been providing advanced features as an "intelligent" support to basic voice communication. This allowed increasing the span of telephony services (i.e. call forwarding, toll free calls, etc.) and differentiating their offers from competitors. The increase of investment from telecommunication operators in their telephony network infrastructure in order to support these services in traditional telephony networks, has witnessed a parallel growth in the use of packet networks, being its use and deployment today almost ubiquitous due to the access to the Internet and Internet Protocol (IP) based applications. It is, at this point, where telecommunication operators need to accommodate the existing infrastructure and investment already made to these new trends and make a non-disruptive transition from their current systems to the new possible architectures. To do that, they should foresee the most effective strategies and solutions for the implementation and provision of the new services, that Internet-based environments allow.

This is the context where this thesis finds its fit, presenting the evolution from traditional services in public switched telephone networks (PSTN)-intelligent network (IN) architectures towards those in hybrid PSTN-IN / IP ones and as an evolutionary step towards totally converged IP networks.

## 1.1. Thesis structure

Chapter 2 presents the initial technical concepts, describing briefly the fundamental architectural structure and signalling mechanisms behind conventional telephony and telephony services. An evolutionary description of the use of Internet and the current state of voice over IP (VoIP) technologies is provided in brief. With that, the chapter establishes a conceptual base for the rest of the thesis.

Chapter 3 presents the two main architectural models allowing today's Internet telephony, H.323 and SIP architectures. The concept of hybrid architecture, as a cooperative merge between conventional PSTN / IN and H.323/SIP architectures follows. After that the chapter deals with the work carried out by the author to provide SIP-based services within a hybrid PSTN-IN / IP telephony architecture developed by the GEMINI consortium. The technology chosen for development, SIP Servlets, is introduced and two basic services, developed for demonstration purposes of the potentials of the GEMINI architecture, are discussed.

Chapter 4 introduces the need for protocol / architecture independent service invocation, current alternatives and how this was achieved, in the context of the GEMINI project, by a web services-based middleware approach. Architectural implications within GEMINI are discussed and a Simple Object Access Protocol (SOAP)-based mechanism, the GEMINI IN (GIN) protocol and resulting GIN API, developed by the author as a network neutral technology for implementation of advanced telephony services, are discussed.

Chapter 5 provides a focus shift to the telephony services discussion. Based on an analogy with information technology realms (corporate networks) and a description on the data storage evolution in those environments, the chapter presents the need for user and service data convergence in future telecommunication networks, as a necessary step for operational convergence, regardless of the level of architectural-network integration. The FlexiNET architecture, an overlay architectural layer in telecommunication networks for data storage and management, is introduced as well as an early work carried by the author within the FlexiNET consortium, to define a data model for user and service profiles.

Finally Chapter 6 provides the closing for the thesis.

# 2. Telephony: Voice and Services

This chapter intends to be an overview to the public switched telephone network (PSTN), presenting its signalling mechanisms and architecture with the Signaling System #7 (SS7) support. Its evolution, from basic voice provision architecture to a service enabled one based on the Intelligent Network (IN) concept and architectural model, are presented afterwards. A brief introduction to service mechanisms in mobile telephony networks is also provided when presenting the Customised Applications for Mobile Networks Enhanced Logic (CAMEL) an augmentation to Global System for Mobile Communications (GSM) networks enabling IN-like service deployment. The means and reasons for IP-based telephony are introduced afterwards.

## 2.1. Conventional Telephony Architecture.

Current public switched telephone networks (PSTNs) are the result of a century evolution from the earliest telephone systems.

The terminals, telephones, connect to the network through the so called central offices (COs) or local exchanges. In the early days of telephony, these COs were manually operated and telephony connections were established by manually switching circuits on a cross connected setup of input and output local lines, with very limited long distance service. The growth in the demand for telephony service made this primitive setup evolve in a two folded way. To improve reacheability between different areas, the interconnection between exchanges was hierarchically arranged. On the other hand the need to cope with a growing number of subscribers in an efficient way, led to the substitution of the manual op-

erator firstly by electromechanical switches, followed afterwards by electronic and digital switches. Today switches are based on specialised computers located within different exchanges in a five-level hierarchy that classifies switch locations as central or local, tandem or toll, primary, sectional and regional exchanges.[2.1][2.2]

PSTN functionality is achieved by the cooperative efforts of two different networks, a circuit-switched trunk network, handling media connections and a packet-switched network handling the signaling that manage these media connections.

ITU's Signaling System #7 (SS7) [2.3] is the packet based signalling network for PSTN. ITU-T Study Group 11 is responsible for the Switching and Signalling (Q.xxx) series of recommendations that define the signalling mechanisms in today's telecommunication networks. In the mid 60's ITU-T, then known as CCITT, developed a digital signalling standard called Common Channel Interoffice Signalling System #6 (CCIS6). CCIS6 later evolved into C7, known as SS7, which has become the signalling standard for the telephony world [2.4].

SS7 defines two different entities, Signaling Points (SPs) sending or receiving signaling messages and Signaling Transfer Points (STPs) routing these messages within the network. Figure 2.1 shows, in green colour, the set of SS7 entities in a basic PSTN network. Due to the reliability requirements for the telephony signalling network, STP entities and signaling links in SS7 networks use to be replicated for redundancy.

SS7 functionality is spread in a layered protocol stack [2.3] as shown in Figure 2.2.

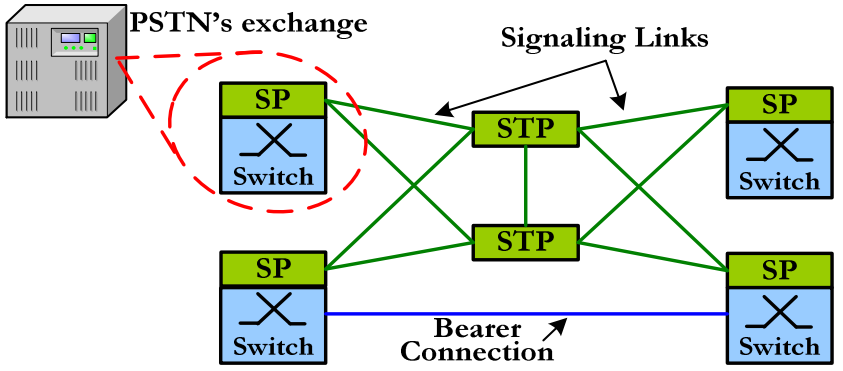



Figure 2.1 SS7's basic entities [2.2].

The SS7 protocol stack is briefly explained as follows: The Message Transfer Part Layer 1 (MTP-1) [2.5] is equivalent to OSI's Physical Layer. The Message Transfer Part Layer 2 (MTP-2) [2.6] is an OSI Data-

Link Network Layer equivalent, providing for signal unit delimitation, alignment and error-detection and correction. It also monitors and reports up to MTP-3, error rates and congestion parameters. MTP Layer 3 (MTP-3) [2.7] is a network-layer process providing functionalities for message discrimination and distribution as well as for traffic management, link management and routing. The Signalling Connection Control Part (SCCP) [2.8][2.9] layer provides addressing mechanisms to route messages to databases and other network entities, acting as a layer-3 protocol for them. The Transaction Capabilities Application Protocol (TCAP) [2.10][2.11] layer provides a way for applications within network entities in the SS7 network to access other applications in a peer-to-peer way, allowing invocation of functionalities hosted in different network nodes, i.e. accessing to a database. The ISDN User Part (ISUP) [2.14][2.15] is the ISDN-oriented augmentation of the Telephone User Part (TUP) [2.12][2.13]. They provide the functionalities for setting up and tearing down the circuits involved in telephone calls in the PSTN [2.4][2.2].

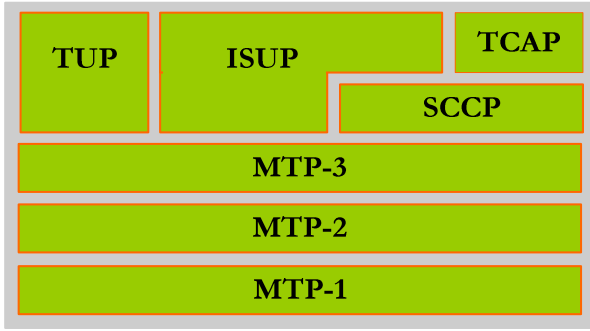



Figure 2.2 Basic SS7's protocol stack.

## 2.2. Telephony Services.

Telephony, as we know it today, is much more than the basic voice service. A number of value-added features, i.e. call forwarding, call waiting, reversed charging, prime-rate numbers, etc, are available in current PSTN networks.

The set of services, as advanced features complementing the basic voice-service, provided by telephony networks is what in the context of this thesis is called telephony services. The architecture that enables the provision of these services is briefly presented in this section.

With the digitization of the switching systems and the integration of computers, operators tried to reduce the maintenance costs of the switches as well as take advantage of the potential this could bring to their networks. The resulting platform could not only provide a basic voice service but could also host and provide a number of advanced features. This would increase the operator's revenue and allow differentiation from competitors in the early deregulated environments of the 80's in USA.

The strategy of programming the network switches to provide telephony services proved soon to be burdensome. All the switches hosting services had to be upgraded in order to host dedicated databases for service-data and program-storage dedicated memories. Furthermore, when service modification was needed, due to errors or simple software upgrades, the modification should be done in all the network switches providing the affected service and in order to provide homogeneous service in the whole network. As a result the service deployment cycle from service conception to deployment was too complex and slow to allow efficient service modifications or simply service deployment.

Trying to cope with these problems by the end of the 80's, the American operator Telcordia developed its "Advanced Intelligent Network" architecture, basically by introducing a new type of entity in its SS7 network, Service Control Points (SCPs), centralizing service and data logic in these elements of the network and delocating it from the switches as shown in Figure 2.3 [2.2][2.4].

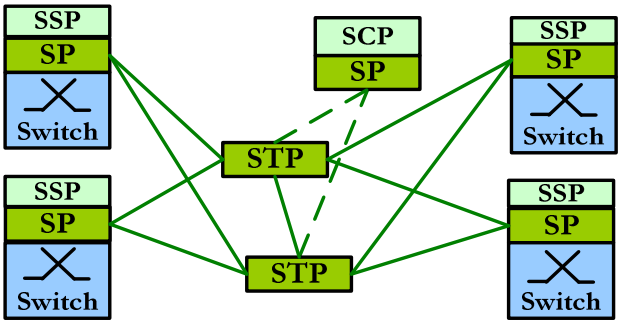



Figure 2.3 Centralizing service logic in a new entity, SCP.

As a result a new functionality in the exchanges, the Service Switching Function (SSF) controlling when and how signaling traffic should be routed towards an SCP in order to invoke a service, was colocated in the switches. Telcordia's successful experience led to a standardization effort to allow interoperability between operators in order to access services from different PSTN networks and between operators and vendors of

equipment. This effort resulted in the first Intelligent Network (IN) standards by ITU-T [2.16].

IN can be seen both as the addition of special nodes to the switching architecture of SS7 and as a software architecture providing special applications, i.e. telephony services [2.2]. Both approaches converge in an orderly manner in the IN Conceptual Model (INCM). The IN Conceptual Model, represents the IN architecture from four different points of view, providing a representation plane for each of them [2.16].

Figure 2.4 displays these four planes and the relations between its elements as described in the following.

The top plane of the INCM is the Service Plane [2.17] describing the services from the point of view of a user: which features the service is made of. These Service Features (SFs) are the basic units of functionality and, like pieces in a blocks-building game, can be presented and combined differently to provide different services. SFs are the subject of standardization of IN, rather than services.

The next plane in a top to down description of the INCM is the Global Functional Plane [2.18]. This plane provides a description of services from an implementator's point of view as an aggregation of software components termed Service Independent Building Blocks (SIBs). The SIBs are the software implementation of one or more service features of the previous plane. A special SIB, the Basic Call Process (BCP), represents the call processing as a finite state machine, the Basic Call State Model (BCSM). The call processing can be interrupted to execute a service program, being the Points of Initiation (POI) the points in the finite state machine where the control is passed to the service logic. When the service logic executes the SIBs it is made of, control is returned to the BCP at the Points of Return (POR).

The next plane of the INCM is the Distributed Functional Plane [2.19], providing a description of the services from the network's point of view as a distribution of functionalities across different entities, functional entities, in the network and the information flows among them. The main functional entities of IN are the following:

- Call Control Function (CCF): provides call processing and control.
- Service Control Function (SCF): hosts the services logic and provides the processing capabilities to execute them, interacting with other functional entities.

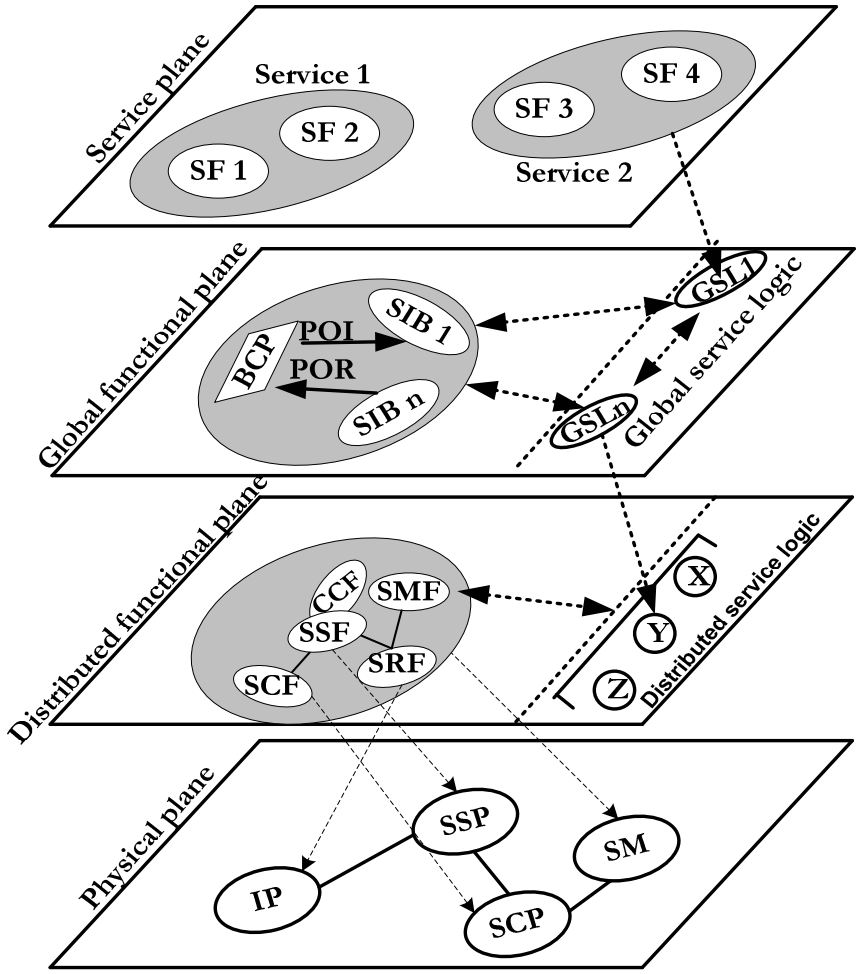



Figure 2.4 Intelligent Network's Conceptual Model [2.16].

- Service Switching Function (SSF): extends the logic of the CCF to include recognition of service control triggers and modify call processing in the CCF as a result of service execution in the SCF. Manages signalling between the CCF and the SCF.
- Service Data Function (SDF): provides database functionality, hosting the service-related data.
- Special Resource Function (SRF): provides specialized resources required for the execution of IN services such us playing announcements, collecting digits, mixing for conferences, etc.

- Service Management Function (SMF): deals with management functionalities like installing and modifying service logic or updating service related data.

The lowermost plane of the INCM is the Physical Plane [2.20], where the network operator's view is provided as a set of physical entities (PEs) hosting the functional entities of the previous plane: Serving Switching Points (SSPs) host CCF and SSF functional entities, Service Control Points (SCPs) host SCF functional entities and frequently integrate also the database systems providing SDF functionality, Intelligent Peripherals (IP) implement SRF functional entities and Service Management Points (SMP) hosts the SMF functional entities. The IN Physical Plane bases and augments the basic SS7 architecture and protocol stack, i.e. a protocol, Intelligent Network Application Protocol (INAP) defines the information flows between SCFs and SSFs or SRFs and is transported over TCAP in the SS7 protocol stack.

As shown in Figure 2.4, a service formed by different service features from the point of view of a user, is implemented as a software application (global service logic) that bases on different reusable software components (SIBs). The set of SIBs interact with the special software component representing the state of a telephone call (BCP). The different pieces of functionality of the application are hosted by different functional entities, each of one located in likely different physical elements in the network.

### 2.2.1. IN evolution.

The standardization of Intelligent Network is a dynamic process. As a result of this process, the IN standard has evolved since the first release, providing increased capabilities. The different releases in the IN standardization time line provide new service capabilities to the network implementing them and due to it they are termed Capability Sets.

Capability Set 1 [2.21], published in 1993, is the starting release. It provides the baseline IN framework and architecture for deployment of advanced, single-ended and single point of control telephony services, with scope in fixed telephony networks and single operators, providing no support for services over operator boundaries. Capability Set 2 [2.22], released in 1997, introduces considerable improvements like foreseeing inter-SCF and inter-SDF communication, allowing service control across different operators networks. Multiparty call models and service logic affecting more than a single BCSM are other new features introduced by CS-2. It also introduces recursivity in the rigid SIB models of CS-1, al-

lowing improved service logic and provides guidelines to overcome feature conflicts between different services. Despite CS-2 is backward compatible with CS-1, the service, global and functional planes in CS-2 are quite more complex and INAP was severely affected [2.2]. Capability Set 3 [2.23], released in 1999, provides a re-specification for INAP as a solution to overcome ambiguities introduced by the modifications caused by CS-2. Capability Set 4 [2.24], released in 2001, updates INAP and the distributed functional plane. CS-4 main new features are those enabling integration of IN operations and Internet, by means of a new functional entity, the Session Manager, providing bridging capabilities between both networks, SS7/IN and Internet based telephony ones and its respective protocols. Other protocols and architectures for integration with Internet telephony systems are introduced also in CS-4, i.e. PINT and SPIRITS. These architectures are described in the following chapter of this thesis.

Due to ambiguities in ITU's CS-1 and CS-2 specifications, most equipment manufacturers and vendors implemented their own INAP flavour, and as a result, interoperability between different vendors' IN equipment, i.e. interoperation of SSPs from vendor A with SCPs from vendor B, was not possible by the mid 90's [2.2]. In order to overcome this problem the European Telecommunication Standard Institute, ETSI, specified ETSI's core INAP [2.25][2.26][2.27][2.28] as a simplified, unambiguous version of the basic INAP functionalities. ETSI's core INAP is a safe choice for current manufactures to assure interoperability of its equipment with other vendors'.

## 2.3. Mobile Telephony and Advanced Services.

Deployment of cellular telephony networks had already started when the IN concept and architecture matured as ITU's CS-1. Despite CS-2 and CS-3 introduced features allowing IN services in mobile networks, ETSI tried to adapt IN to the specific characteristics of GSM networks. The resulting effort was ETSI's Customized Applications for Mobile Enhanced Logic (CAMEL)[2.6].

CAMEL architecture resembles heavily IN's architecture. Main differences come from a service execution point of view, where CAMEL differentiates whether the service has been triggered from the GSM network the subscriber belongs to, the home network, or from a different one, the visited network, while roaming there. CAMEL services are always executed in the subscriber's home network. CAMEL introduces, as applica-

tion protocol for communication between SCFs and SSFs or SRFs, an adapted version of the INAP protocol called CAMEL Application Part protocol (CAP). Special purpose databases, the Home and Visitor Location Registers, keeping user vital information for the operation of the GSM network are accessed also by SCF entities via a dedicated protocol termed Mobile Applications Part (MAP).

Figure 2.5 gives a simplified view of a CAMEL network as a basic GSM network were the mobile switching center (MSC) functionality has been enhanced with a SSF entity allowing the triggering of services hosted in a SCF functional entity and communicating with it via CAP. Other functional entities such as SRFs or SDFs have been omitted for simplicity.

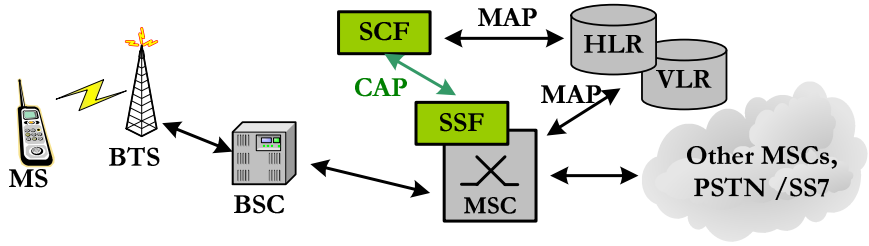



Figure 2.5 CAMEL elements as an overlay on a basic GSM network.

# 2.4. The advent of Internet Telephony.

The provision of real-time voice transport service over packet networks and the architectures and signalling mechanisms that make it possible are not new at all: [2.29] documents the Network Voice Protocol, implemented back in December 1973 as part of the Network Secure Communications project of the US Advance Research Projects Agency (ARPA). At that time, Arpanet was formed by 23 hosts at military and civil research locations and the term Internet, as a network of independent subnetworks was just coined and ready to be presented by Cerf and Kahn with the initial TCP/IP protocol definition five months later [2.30]. During the following decade, Internet and its underlying mechanisms matured as the realm of military and academic research institutions and activities. It was not until 1989, when Tim Berns Lee at CERN implemented a hypertext system to provide efficient information to the high-energy physics community [2.31], that the Internet jumped out the specialised research environments and its use generalised via the World Wide Web (WWW) and harmonized utilization of web browser applications. In just a decade and

fostered by the adoption of broadband access systems, the Internet become an environment ubiquitely pervading every realm of our lives.

Internet protocol (IP), provides a technology independent / content neutral mechanism of transmitting digitalized information, regardless of its source, destination or media characteristics. Voice is just one more of these media variants and voice over IP (VoIP) is one of the applications of this technology.

From an operator's point of view the transmission of digitalized voice over IP packets maximizes the efficiency of its data network. There is no need to maintain a separate network dedicated for voice, with different technology and its derived maintenance costs. The operator can reuse its data infrastructure to provide the basic voice service, at increased capacities and reduced prices, since the use of audio codecs allow reducing the needed bandwidth per voice channel and the statistical multiplexing within the network maximizes its utilization. For the user, this is reflected in a reduction in the overall price for the call with, hopefully, not substantial quality degradation. The increased and increasing broadband capabilities of the access networks with the proper QoS assurance mechanisms within the core network, allow today minimizing the common degrading effects of transmitting real time voice over IP (delay, jitter and packet lost) using specific techniques for transport of real time data over IP [2.32][2.33][2.34][2.35][2.36].

But, as in the case of conventional telephony, IP Telephony is more than the basic voice service. IP Telephony has grown not only as a multimedia communication platform but as a solution for advanced (VoIP) telephony services far beyond those initially envisioned by Intelligent Network and thanks to the synergies that Internet allows. Multiconference calls, with video communication, buddy lists with presence information, file sharing and even collaborative working spaces are possible today with basic applications in almost every desktop computer. IN-like services incorporating these different media, and taken advantage of web-based interfaces for management and customization, as well as the powerful software platforms allowing them, are the next wave of telephony services to come.

While first VoIP initiatives were far from commercially blossoming, operators felt confident about their traditional business models where telephony services were provided via PSTN-IN based networks. The still limited use of Internet in home environments, the narrowband capabilities offered by the access networks and the race for the mobile cellular systems deployment and market share made them feel confident about their positions, neglecting the impact of VoIP.

The situation today is rather different. Broadband deployments in the access networks allow real-time delivery of data over IP and the public Internet with voice quality comparable to that of the PSTN. Internet is a ubiquitous tool, accessible in most households, and users use it for any of their daily activities. Considering Internet as an alternative to the traditional telephone service is not just a quest for a set of users but a mere new use for the common tool Internet has become. Furthermore, when the costs are comparatively lower (when not inexistent) the quality becomes comparable and the broadness of services offered considerably attractive. While VoIP solutions start being attractive, several hurdles have to be still overcome to guaranty its future success: quality of service assurance, access to emergency services, call location & tracking capabilities, inter-operator agreements, protocol and architectures maturity and national as well as international regulation.

Nevertheless, today the future, though still distant, looks quite bright for Converged Architectures where IP is the solely underlying technology. Operators that do not acknowledge it will probably disappear in the long term, since the subscribers will demand to adapt their communication experiences to the new environments and possibilities and will migrate their subscriptions to those environments satisfying their requests. As a transitional step towards these converged environments and as a way to maximize the return of investment of the already deployed infrastructures, operators are starting to offer telephony services through hybrid networks, where traditional PSTN/IN and Internet Telephony architectures coexist and interwork.

## 2.5. Summary

This chapter has presented the technical point of departure for this thesis, introducing the concept of telephony services and its evolution from conventional telephony to Internet telephony. The rest of the thesis, bases on the terminology and concepts presented herein.

## 2.6. Chapter References


[2.1] "Understanding Telecommunications", Ericsson Telecom, Telia and Studentlitteratur 1998.

[2.2] J.Zuidweg, "Next Generation Intelligent Network", Artech house Inc. 2002.

[2.3] ITU-T Recommendation Q.700, "Introduction to Signalling System #7", 1993.

[2.4] T.Russell, “Signaling System #7”, McGraw-Hill Telecommunications 2000. ISBN: 0-07-136119-7.

[2.5] ITU-T Recommendation Q. 702, “Signaling Data Link”, 1989.

[2.6] ITU-T Recommendation Q. 703, “Signaling Link”, 1996.

[2.7] ITU-T Recommendation Q. 704, “Signaling Network Functions and Messages”, 1996.

[2.8] ITU-T Recommendation Q. 711, “Functional description of the signaling connection control part”, 2001.

[2.9] ITU-T Recommendation Q. 712, “Definition and function of signaling connection control part messages”, 1996.

[2.10] ITU-T Recommendation Q. 771, “Functional description of transaction capabilities”, 1997.

[2.11] ITU-T Recommendation Q. 772, “Transaction capabilities information elements definition”, 1997.

[2.12] ITU-T Recommendation Q. 721, “Functional description of the SS7 Telephone User Part”, 1988.

[2.13] ITU-T Recommendation Q. 722, “General functions of telephone messages and signals”, 1988.

[2.14] ITU-T Recommendation Q. 761, “SS7-ISDN User part functional description ”, 1999.

[2.15] ITU-T Recommendation Q. 762, “SS7-ISDN User part general functions of messages and signals”, 1999.

[2.16] ITU-T Recommendation Q. 1201, “Principles of Intelligent Network Architecture”, 1993.

[2.17] ITU-T Recommendation Q. 1202, “IN Service Plane Architecture”, 1997.

[2.18] ITU-T Recommendation Q. 1203, “IN Global Functional Plane Architecture”, 1997.

[2.19] ITU-T Recommendation Q. 1204, “IN Distributed Functional Plane Architecture”, 1993.

[2.20] ITU-T Recommendation Q. 1205, “IN Physical Plane Architecture”, 1993.

[2.21] ITU-T ITU-T Recommendation Q. 1211, “Introduction to Intelligent Network Capability Set 1”, 1993.

[2.22] ITU-T ITU-T Recommendation Q. 1221, “Introduction to Intelligent Network Capability Set 2”, 1997.

[2.23] ITU-T ITU-T Recommendation Q. 1231, “Introduction to Intelligent Network Capability Set 3”, 1999.

[2.24] ITU-T ITU-T Recommendation Q. 1241, “Introduction to Intelligent Network Capability Set 4”, 2001.

[2.25] ETSI ETS 300 374-1, “IN CS1 Core INAP Protocol Specification”, 1994.

[2.26] ETSI EN 301 140-1, “IN CS2 Core INAP Protocol Specification”, 1999.

[2.27] ETSI EN 301 931-1, “IN CS3 Core INAP Protocol Specification. Common Aspects”, 2001.

[2.28] ETSI EN 302 039-2, “IN CS4 Core INAP Protocol Specification. SSF-SCF Interface.”, 2002.

[2.29] IETF RFC 741, “Network Voice Protocol (NVP)”. November 1977.

[2.30] V.G Cerf, R.E. Kahn, “A protocol for packet network intercommunication.”, IEEE transactions on Communications, Vol Com-22, May 1974.

[2.31] http://en.wikipedia.org/wiki/World_Wide_Web

[2.32] H.Schulzdrine & alt., IETF’s RFC 1889, “RTP: A transport protocol for real time applications.”, January 1996.

[2.33] S.Casner, V. Jacobson, IETF’s RFC2508, “Compressing IP/UDP/RTP Headers for Low-Speed Serial Links”, February 1999.

[2.34] T.Koren et al., IETF’s RFC 3545, “Enhanced Compresed RTP for links with high delay, packet loss and reordering”, July 2003.

[2.35] B.Thompson et al., IETF’s draft-ietf-avt-tcrtp-08, “Tunneling Multiplexed Compressed RTP”, February 2005.

[2.36] E. Rosen et al., IETF’s RFC 3031, “Multiprotocol Label Switching Architecture”, January 2001.

# 3. Hybrid Telephony Architectures and Service Deployment

While the signaling mechanisms in conventional telephony are widely standardized and mature to allow interoperability among different PSTN networks, both for connection management and service deployment, this has not been the case in VoIP environments due to the newness of deployment initiatives and the lack of maturity of the enabling protocols.

In this chapter the evolutionary path in VoIP signaling is presented and details of protocols and architectures enabling service deployment in hybrid architectures, those combining PSTN/SS7/IN and IP-based infrastructures, are provided. Initially the two main protocols for VoIP signalling are presented, H.323 and SIP, with their respective architectures and their potential for service deployment. After that, different initiatives enabling hybrid telephony architectures are presented: the means of transport of SS7/IN signalling over IP protocol and the mechanism to translate traditional telephony addressing into one based on domain name and user name. Two seminal architectures, PINT and SPIRITS, demonstrating interoperation between PSTN/SS7/IN networks and SIP based networks are presented afterwards, as well as the integration of a SIP based infrastructure within the Universal Mobile Telecommunication System (UMTS) network in order to allow deployment of IP-based multimedia services.

GEMINI, a hybrid architecture for telephony services interoperation is introduced. A SIP-based application server within the GEMINI architecture, its enabling technology and demonstration services the author designed, developed and deployed on it are finally presented. The work described herein was presented by the author in [3.1][3.2].

# 3.1. H.323

ITU's H.323 v.1 [3.3] was released in December 1996 as a standard for a real-time video telephony system over non-guaranteed quality of service local area networks (LANs). Since then, four more versions of the standard have been released. H.323 v.5 [3.4] is defined as a standard for packet-based multimedia communication systems.

H.323 defines a set of functional entities:

- *Terminals* that terminate media and signaling for end users.
- *Multi-point control units* (MCUs) that enable conferencing among multiple terminals.
- *Gateways* that allow interoperation with other networks by means of conversion between different signaling protocols and media formats.
- *Gatekeepers* that provide authentication, authorization and accounting, address resolution and routing services to entities in their domain.
- *Border elements* that provide public access to administrative domains for access authorization or address resolution.
- *Servers* dedicated to specialized functionality such as supplementary services (*feature servers*).

A view of the set of protocols which compound the H.323 umbrella is depicted in Figure 3.1.

For real-time media transport, ITU-T adopted for the H.323 specification the industry accepted protocols standardized by IETF, namely RTP and RTCP [3.5].

H.225.0 [3.6] defines the signaling protocol to set-up and release a call. It is derived from ISDN's Q.931 [3.7]. H.225.0-Registration, Admission and Status (RAS) defines the signaling protocol for the communication between endpoints and gatekeepers. Annex G of H.225.0 defines the signaling between border elements in different administrative domains. H.245 [3.8] defines the signaling protocol for endpoints (terminals, gateways or MCUs) to exchange capabilities and control media streams.

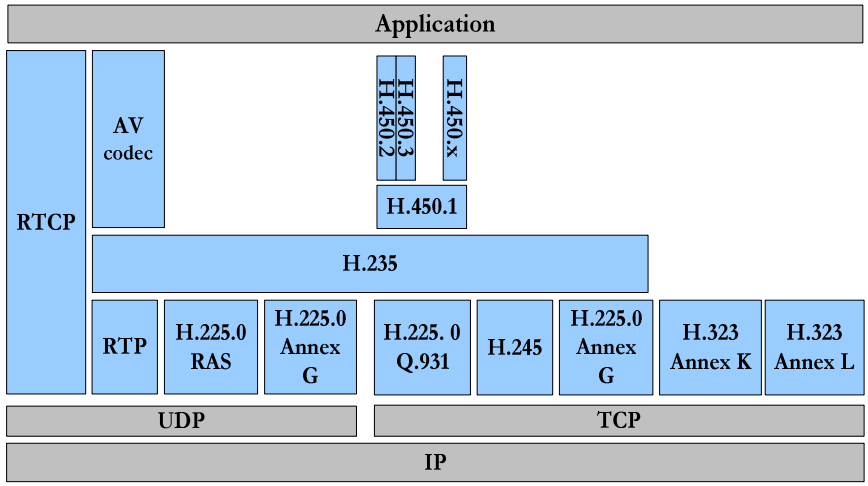



Figure 3.1 H.323 protocol suite.

An example of H.323 basic call set-up mechanisms appear in Appendix 1.

H.235 [3.9] defines the security mechanisms for authentication, nonrepudiation, confidentiality and data integrity in H.323 networks.

All H.323 protocol data units are specified and coded with ASN.1 notation [3.10]. In order to avoid the growth and modification of the ASN.1 definition for the H.323 protocols data units, a generic extensible framework was added to H.323v4 [3.11]. A number of new features have been added in H.323v5 [3.4] based on these mechanisms and under the H.460.x umbrella.

H.323 provides three different mechanisms for service provision as detailed in the following section.

### 3.1.1. H.323 and Services

H.450 series defines a framework for deployment of supplementary services in a H.323 architecture. H.450.1 [3.12] defines the generic signaling mechanisms necessary for the execution of supplementary services without intervention of network entities apart from endpoints and feature servers. A number of service specific recommendations have been produced as displayed in Table 3-1.

| Recommendation | Service |
|---|---|
| ***H.450.1*** | **Generic functions** |
| ***H.450.2*** | **Call transfer** |
| ***H.450.3*** | **Call diversion** |
| ***H.450.4*** | **Call hold** |
| ***H.450.5*** | **Call park / Call pick up in** |
| ***H.450.6*** | **Call waiting** |
| ***H.450.7*** | **Message waiting indication** |
| ***H.450.8*** | **Name identification** |
| ***H.450.9*** | **Call completion** |
| ***H.450.10*** | **Call offering** |
| ***H.450.11*** | **Call intrusion** |
| ***H.450.12*** | **Common information** |

Table 3-1 H.450 standard services

Apart from detailing the common, generic functions for supplementary services, H.450.1 defines the procedure to handle non supported H.450 data units. In such a way, interoperability between endpoints with different feature sets is assured and progressive service deployment across networks allowed. H.450.12 [3.13] further supports this aim allowing the exchange of feature capabilities among endpoints. A generic extensibility framework was added in version 4 of the H.323 specification providing also a mechanism for feature negotiation.

H.323 Annex L, Stimulus Feature Control defines the signaling mechanisms between a dumb terminal and a centralized feature server in charge of controlling the features at the endpoints. The terminal simply provides stimuli corresponding to the service to be executed.

H.323 Annex K, Application-Layer Feature Control defines an alternative service plane above H.323's control plane so that service control sessions over HTTP can be established after the general H.225 procedures. These sessions can be used to access service logic through HTML pages.

## 3.2. SIP and Service Provisioning.

Session Initiation Protocol, SIP is an application layer control protocol specified by the Internet Engineering Task Force (IETF) in 1999 [3.14] to create, modify and terminate sessions with one or more participants. Based on other successful protocols such as HTTP and SNMP, the initial simplicity of this text based protocol fostered its adoption and deployment by the Internet community as signalling mechanism for VoIP sessions. Updated in 2002 [3.15], SIP provides basic call signalling functionality as separate transactions, where a transaction is defined as a sequence of a request and its responses. The description of the media streams within the session is not provided by SIP. IETF suggests the Session Description Protocol [3.16] for this aim, and as content of the SIP message body at the initial messages starting the session.

SIP entities are basically clients and servers, where a client is an entity generating a request and a server an entity generating responses. Registrars are functional entities that bind the e-mail like SIP identifier of a user client with its IP address at the time of registration in the SIP network. Special servers, proxy and redirection servers, provide a location service by consulting the current location of users in a registrar and providing the proper routing services of the requests towards the location of the request target. An example of SIP basic operation appears in Appendix 2.

SIP responses are grouped in ranges according to its semantic content and in 6 different groups: informational, success, redirection, client error, server error and global failure.

SIP defines six basic request types, *REGISTER* to inform a registrar about the current location of a user, *INVITE* to initiate a session and transport the session description, *ACK* to acknowledge the reception of a final response, *CANCEL* to cancel pending transactions, *BYE* to abandon a session and *OPTIONS* to query a server about its capacities. The basic set of six requests, provides a global scale of interoperability for common-basic operations.

Soon this set of requests proved to be non-sufficient to perform other necessary operations such as providing information during a session without affecting the state of the session, indicating caller preferences, provide asynchronous notification of events or allowing session transfer or third party call control.

In order to solve these and other shortcomings, enhancements to the SIP core protocol have been defined in the form of new requests, new header

or header fields into existing ones and/or new event packages. This has increased the initial complexity of SIP based systems as a trade-off to broaden its applicability.

To solve interoperability problems, SIP incorporates a negotiation mechanism (Require/Supported headers) that allow agreement on the extensions used in any given SIP session. This mechanism allows to identify the extensions supported by the participants in a session and to determine those that will be used for it. A list of the approved SIP extensions, by December 2004, appears in Appendix 3.

Neither SIP nor any of its accompanying documents specify services, like it is the case for H.323 with the H.450.x series. SIP provides the functionality over which to build services without constraining the developer to a model for the service behaviour.

This apparent anarchy in the service layer is one of the strongest points for SIP, allowing flexibility for service developers and extensibility of the protocol behaviour to satisfy functional needs of specific services.

### 3.2.1. H.323 vs SIP

Despite H.323's H.225 signalling is easier to interoperate to PSTN/SS7 signalling, the binary encoding of H.323 messages, its complexity (number of different protocol-components) and lack of flexibility (highly constrained standardized framework) affected H.323 acceptance once SIP gained maturity. The simplicity and light-weight of SIP, its text-based encoding easing debugging (though reducing its transmission efficiency), and its generality, based on the lack of constraining applicability specifications, made soon SIP the preferred protocol for VoIP and multimedia conferencing in Internet. These factors, together with its scalability, flexibility and extensibility have made it also the preferred protocol for the so-called Internet Multimedia Subsystem [3.17] for Universal Mobile Telecommunication Networks (UMTS) described in the following section [3.18][3.19][3.20][3.21].

## 3.3. Hybrid Architectures

Hybrid Architectures, those based on interoperation between PSTN/IN an IP-based infrastructures, are a necessary step in the architectural evolution of telephony service provision. They are the natural way for incumbent operators to compete with those new players whose architectures are solely based on packet technology, the so-called All-IP or Converged

Architectures and a transitional phase allowing them deployment of IP-based telephony services. By reusing a parallel packet-based network, the proper interoperation mechanisms with the legacy PSTN/IN infrastructures and the creation of hybrid services is possible. Meanwhile, the legacy infrastructure is still in use and revenues are assured for the operator. Different approaches toward hybrid solutions have been subject of study and standardization.

Within the Internet Engineering Task Force (IETF), due to its flexible and open mechanisms, soon solutions towards interoperation between PSTN/IN and IP domains were proposed.

The Signalling Transport (SIGTRAN) working group of the IETF addresses the transport of packet-based PSTN signaling over IP Networks, taking into account functional and performance requirements of the PSTN signaling. SIGTRAN defines the Stream Control Transport Protocol (SCTP) [3.22] and adaptation layer specifications for the transport of PSTN signalling protocols over SCTP: M2UA [3.23] for MTP2, M3UA [3.24] for MTP3, SUA [3.25] for SCCP. SCTP arises from the need to transport a variety of PSTN protocols over IP networks, solving the problems that existing IP transport protocols (TCP, UDP) face on doing so: while UDP provides a non tolerable unreliability for the transport of information such as telephony signalling, TCP presents security vulnerabilities, i.e. SYN flooding attack , and a head-of-line-blocking (HOL) problem[1] that might affect the delay and resilliance in the operation of a signalling network based on it [3.26][3.27].

SIGTRAN proposals allow using IP networks as transport mechanism for the SS7 or IN signalling as displayed in Figure 3.2. In the figure, a SS7/IN service switching point (SSP) communicates via INAP with an IP-based service control point (SCP) thanks to SCTP and the adaptation layer defined for users of SCCP as defined by SIGTRAN.

Carrying SS7/IN signalling over SIGTRAN can be used by an operator to alleviate signalling traffic from its conventional network (PSTN or PLMN) by means of diverging signaling traffic towards an IP-based network in the event of pick demands or failure in network segments.

[1] While devoting a TCP connection per signalling flow is impractical, the HOL problem might affect the performance of a set of signaling flows, multiplexed within a TCP connection, in the event of a problem affecting one single flow of those multiplexed. With SCTP, the HOL problem is reduced to the scope of the individual flows within a SCTP association.

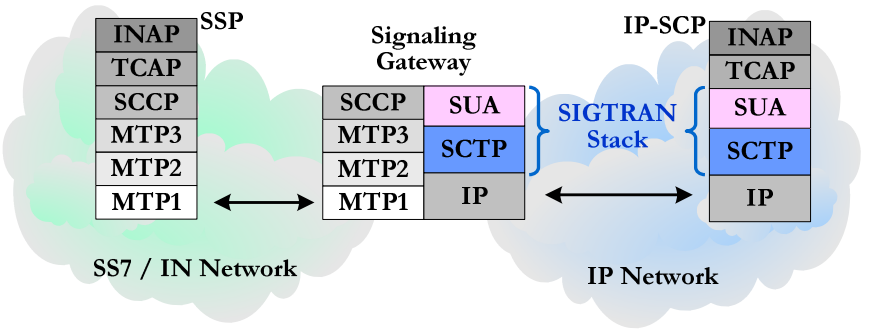



Figure 3.2 IN signaling over IP and SIGTRAN stack.

Also, signaling over SIGTRAN can be used as a way of joining independent or geographically unconnected PSTN/SS7 networks, without the need of investment in dedicated infrastructure when an IP infrastructure already exists, and can be used, among them.

Another study within the IETF towards hybrid PSTN/IP interactions is carried out by the Telephone Number Mapping (ENUM) working group of the transport area. ENUM defines [3.28] the means by which an E.164 number of the PSTN domain [3.29] is expressed as a fully qualified domain name [3.30][3.31] in a specific internet infrastructure domain defined for this purpose.

The Session Initiation Proposal Investigation (Sipping) working group of the IETF deals specifically with applications of SIP to different problems related to telephony and multimedia and enhancements to SIP to fulfil the different needs. As a result of the work carried out for telephony applications, different SIP-PSTN interoperation mechanisms have been studied and mapping between ISUP and SIP specified [3.32][3.33][3.34][3.35].

The Services in the PSTN/IN Requesting Internet Services (SPIRITS) Working Group of the IETF Transport Area deals with the mechanisms by which IP-based services can be triggered from PSTN/IN entities, as well as the protocol mechanisms through which PSTN can request actions to be carried out in the IP network in response to events (IN Triggers) occurring within the PSTN/IN. The SPIRITS Architecture [3.36] was born as a response to a previous work in support of the services initiated in the reverse direction - from the Internet to PSTN [3.37]: The Public Switched Telephone Network / Internet Inter-Networking (PINT) services combined the Internet and PSTN services in such a way that the Internet is used for non-voice interactions, while the media (voice and fax) are carried over the PSTN. The signalling interaction between a PINT domain and a PSTN domain is shown in Figure 3.3.

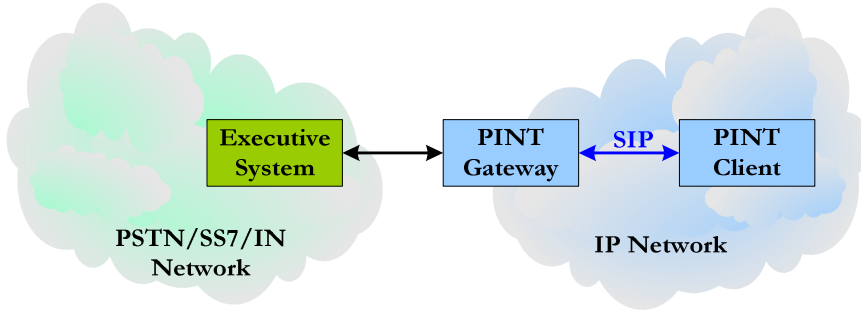



Figure 3.3 PINT architecture and interaction with PSTN

SIP is used within PINT to carry requests for telephone service from the PINT client to the PINT gateway. PINT clients and servers are just SIP clients and servers. The PINT Gateway that appears to a SIP system as a SIP server, must communicate to a telephone system (PSTN/IN) entity, termed Executive System. The means of communication from the PINT Gateway to the Executive System in the PSTN/IN is not clarified. This interface is implementation dependent.

On the other hand, SPIRITS aims to support services that originate in the PSTN and require the interactions between the PSTN and the Internet. As shown in Figure 3.4 the SPIRITS client is colocated with the IN Service Control Function (SCF) that behaves also as Executive System for the PINT architecture.

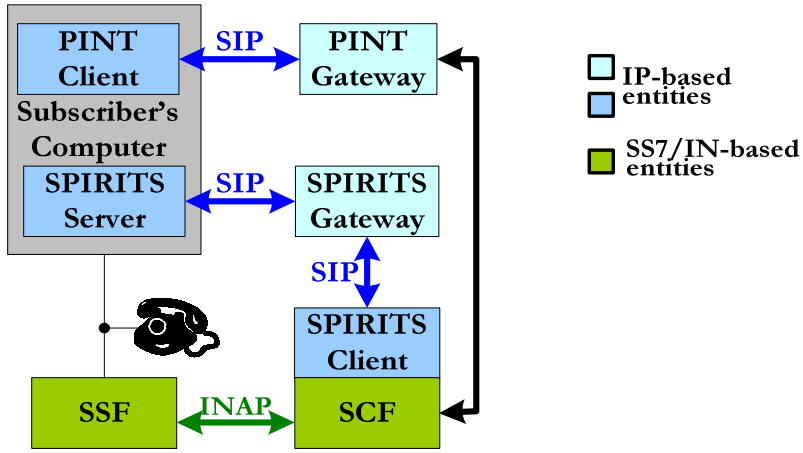



Figure 3.4 Combined PINT and SPIRITS architecture.

Target services for the SPIRITS architecture assume that IP connectivity is achieved by the subscriber via a dial-up connection through its telephone line. The SPIRITS services are invoked when a message from the

SPIRITS client arrives to the SPIRITS gateway. After processing the message, the gateway passes it to the SPIRITS Server. Communication between SPIRITS elements is based on subscription and notification SIP mechanisms.

Main service examples are Internet Call Forwarding, where a call to the telephone terminal is forwarded to another number if the line is busy during connection to Internet, Internet Call Waiting where a call to the subscriber's telephone line is set on hold while the subscriber is connected to Internet, the subscriber is notified of it and decides how to handle it (forward to another phone number, to a VoIP connection, voice mail or close the Internet connection to answer it). It is obvious today, with the existence of ADSL connections or HFC enabled, that these services are outdated.

Specific work has also been carried to define the proper interaction mechanisms between SIP and INAP, allowing the interworking of networks and services based on these protocols, as in the previous architectures: early in the development stages of SIP, a study of implementation of IN services via SIP was performed in [3.38]. Mechanisms for interoperation between INAP and SIP were firstly analyzed in [3.39][3.40][3.41][3.42], works summarized and extended in [3.43].

The European Telecommunications Standard Institute has carried also work towards hybrid networks and service architectures within their technical committees for Services and Protocol for Advanced Networks (SPAN) and Telecommunication and Internet Protocol Harmonization Over Networks (TIPHON). In 2003 SPAN and TIPHON merged to the technical committee Telecoms and Internet Services and Protocols for Advanced Networks (TISPAN) [3.44].

The 3rd Generation Partnership Project (3GPP) [3.45] has defined an architecture for IP-based service deployment as the "IP Multimedia Core Network Subsystem (IMS)" [3.46], a fundamental part of Universal Mobile Telephony System (UMTS) network architecture that allows the deployment of multimedia services. This architecture is based on SIP.

A simplified diagram with the main functional entities and interfaces of this IMS service architecture appears in Figure 3.5. As displayed in that diagram, the central entity is the Call Session Control Function providing registration to endpoints accessing the system and proxying of SIP signaling to / from the application servers. These host SIP-based service logic or bridge to other service provisioning environments, i.e. CAMEL or OSA (introduced in the following chapter). The Home Subscriber Server is a data base accessible from the different servers maintaining the unique user profile for each end user.

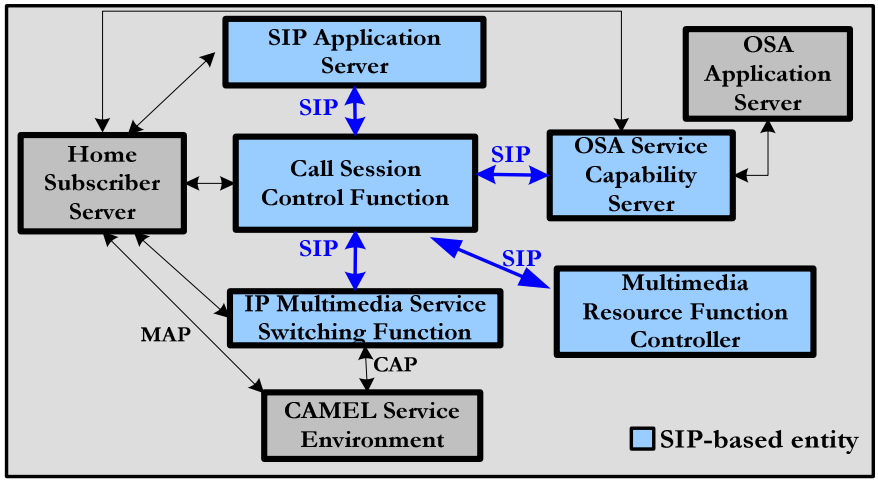



Figure 3.5 Service Architecture in 3GPP's IMS

The functional elements of the IMS require the provision of IP-connectivity through the UMTS core network, via the network's Serving GPRS support node (SGSM) and Gateway GPRS support node (GGSN), with UTRAN as radio access network towards/from the UMTS terminals.

# 3.4. GEMINI Project

The European IST project Generic Architecture for customised IP-based IN services over hybrid Voice over IP and SS7 (GEMINI), was launched in spring 2002 and it ended successfully in spring 2004.

Partners contributing to the project were Alcatel SEL (D), Intracom (EL), Solinet (D), Telekom Austria (AT), Otenet (EL) and Research Center COM (DK).

The main objective of GEMINI was to offer, in an IP based environment, existing and next-generation customized and personalized IN services. Additionally, the proposal focused on the interworking between IP- and SS7- based architectures in order to deliver value-added and extended IN services in a hybrid environment to both PSTN and VoIP clients.

Achieving this goal required the design of a modular and scalable architecture for the provisioning of IN/IP-based services and the definition of all the elements of this architecture in terms of entities, protocols and interfaces. Many of the technologies presented so far are involved in the GEMINI architecture as it is explained in the following.

GEMINI architecture is based on SIP protocol as the signaling mechanism to access the services provided by the IP side of the hybrid architec-

ture. This does not preclude the existence of H.323 based infrastructure since the proper interworking mechanisms were introduced to allow VoIP end user diversity. Services in the IP side of the architecture are accessed by VoIP terminals as well as from conventional PSTN terminals from the PSTN/SS7 portion of the architecture. Reciprocally, IN services in the PSTN/SS7 portion of the architecture are accessed by VoIP terminals from the IP side. The interworking between both sides is allowed by a gatewaying system. The GEMINI architecture is displayed in Figure 3.6 and explained in the following.

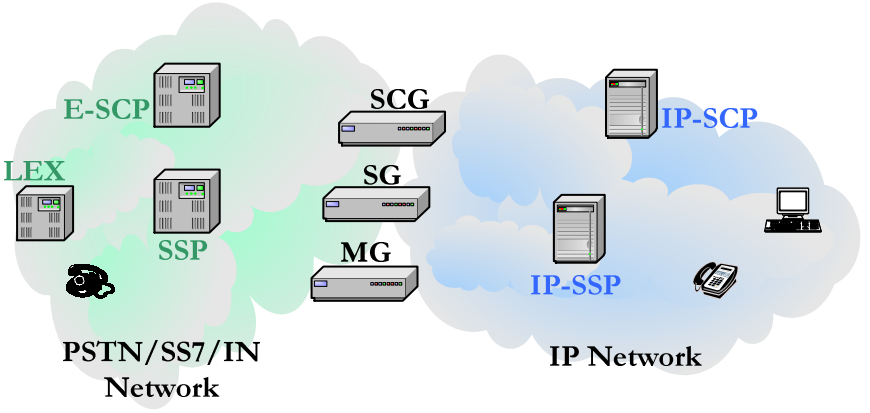



Figure 3.6 GEMINI Architecture

The PSTN/SS7 side of the architecture is a conventional PSTN/IN telephony network as described in Chapter 2 of this thesis. A Service Control Point (SCP) hosting IN's service logic was enhanced (E-SCP) to allow service customization from the IP domain through a Web interface.

The IP side of the architecture was created following a functional replication of the SS7/IN entities. While an entity termed IP-based Service Control point (IP-SCP) hosts the IP-based service logic, an entity termed IP-SSP performs the proper routing of signalling within the VoIP network as well as triggers the IP-based service execution by routing signalling messages towards the IP-SCP. For practical reasons the data repositories for the service logic were also integrated in the IP-SCP. The main signalling mechanisms within the IP side of GEMINI's architecture are SIP-based. Deployment of SIP-based IN-like services in real telephony environments was a challenge at the time GEMINI was devised and its demonstration one of the interests of the GEMINI project. The IP-SCP behaviour corresponds to the one of a SIP-based Applications Server. Further descriptions of the IP-SCP and the IP-based service logic are provided in the following sections of this chapter.

The IP side of GEMINI's architecture can be compared with the service architecture provided by the 3GPP's IP multimedia Subsystem of UMTS. The SCCF functionality within the 3GPP's IMS is provided by the IP-SSP in GEMINI and the HSS functionality is integrated with that of the Application server in the IP-SCP.

To allow a broader diversity of users and reuse of existing H.323-based VoIP deployments, the existence of H.323 terminals was allowed by means of integrating an H.323/SIP conversion entity within the IP-SSP. A conference bridge (H.323's multiconference unit) was also integrated in the IP-SSP entity.

A gatewaying system, termed IN Services Gateway, provides interworking between both sides of the GEMINI architecture through the cooperation of three different functional entities:

- Signaling Gateway (SG): provides interworking between call control signaling entities in the PSTN/SS7 network and the ones in the IP domain by means of remapping between ISUP over MTP and ISUP over M3UA.
- Service Control Gateway (SCG): provides interworking between service switching and control entities in the PSTN/SS7 network and the ones in the IP domain by means of remapping between SCCP over MTP and SCCP over M3UA, allowing INAP messages over TCAP and SCCP to be used in an IP environment.
- Media Gateway Function: provides the transformation mechanisms necessary to adequate the media flows from one side to the characteristics of other.

The signalling paths involving these entities appear in Figure 3.7.

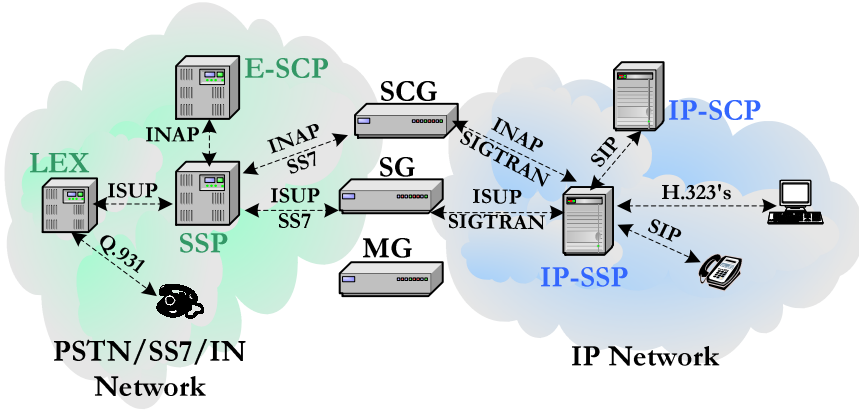



Figure 3.7 Signalling in GEMINI architecture

As displayed in the figure, the gatewaying system performs a SS7/SIGTRAN mapping while the protocol conversion to SIP happens in the IP-SSP.

Services implemented in the IP side of the GEMINI architecture can be accessed from conventional telephones in the PSTN side. This access to the IP-based service logic from the PSTN/SS7 side of the architecture involves the following operations: a telephone terminal initiates a session setup towards its serving local exchange via Q.931. The interexchange signalling mechanism based on SS7's uses ISUP to access a serving SSP. Two alternatives are possible at this point:

- The SSP can route the ISUP request towards the signalling gateway set, which appears for the SSP in Figure 3.7 as the next signalling point in the SS7 network. In the gatewaying set, the ISUP message will be transmitted over SIGTRAN towards the IP-SSP by the SG entity. There, after an ISUP/SIP mapping, a SIP request is sent towards the IP-SCP where the service is executed. Any response towards the initiator follows opposite mechanisms over the same path.
- A detection point for the service number has been pre-established in the SSP and upon reception of the ISUP message an INAP message is triggered from the SSP towards the gatewaying set that is seen by the SSP as the proper SCP for this service. In the gatewaying system, the SCG would relay the INAP flow over SIGTRAN towards the IP-SSP where the INAP/SIP conversion would finally produce the SIP signalling towards the IP-SCP, where the service logic would be executed. This option was finally not tested in GEMINI since no INAP handling capacities were finally developed in the gatewaying system (SCG) neither in the IP-SSP.

The GEMINI architecture adds an unnecessary level of complexity, since the protocol conversion could have been performed in an earlier single point, the gatewaying system as shown in Figure 3.8, instead of carrying the IN/SS7 signalling towards the IP-SSP. The adopted solution was due to the interest of the involved partners on introducing a SIGTRAN flow within the architecture, reusing in such a way previous developments.

The invocation of IP-based services by IP based terminals is straightforward, with the role of the IP-SSP as a mere proxy server for SIP based terminals and as a protocol converter and SIP proxy server in the case of H.323 terminals.

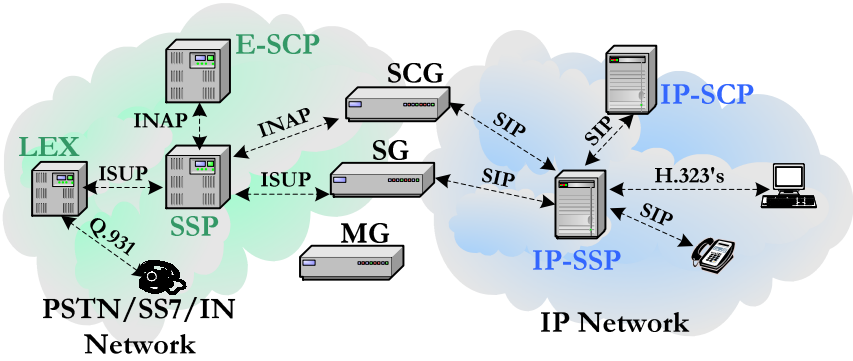



Figure 3.8 GEMINI's alternative architecture

# 3.5. GEMINI's IP-based Service Logic Technology

As commented previously, it was one of GEMINI's objectives to demonstrate SIP-based IN-like service deployment. By April 2002, when the GEMINI consortium started the specification process for the GEMINI architecture, a new software technology aimed to support applications communicating by means of SIP, was publicly released for review by the Java Specification Request 116 Group [3.47]. This technology, termed SIP Servlets, was chosen as the base for implementing the IP-based services of the GEMINI architecture, due to its characteristics and capabilities to provide a system-independent (Java based) tool for creation of powerful applications capable of integration with other environments, i.e. Java-based API's for database access, networking, etc. A brief introduction to the technology is presented in this section.

## 3.5.1. Server-side Java Applications: Servlets

A servlet is a Java class (application) that allows extending the capabilities of a server to host applications accessed via request-response models. Servlets are to servers what applets are to clients [3.2].

Java Servlets were born as an alternative to Common Gateway Interface (CGI) technology and as a way to overcome the problems of platform dependency and lack of scalability that this technology had [3.48].

The javax.servlet package [3.49] contains a number of classes and interfaces that describe and define the relation between a servlet class and the

runtime environment provided for an instance of such a class by a conforming Servlet Container. The Servlet Container is an application monitoring a port on a given IP address of the server where it runs in. When a request arrives to the server, at the port being monitored, the Container parses the request and puts its information in a Servlet Request object that is passed to a servlet. In the reverse path, the response parameters are bundled by the Container in a Servlet Response object. The main role of the Servlet Container is to perform routing of requests towards applications, selecting the Servlets to invoke and its order. To allow this, there is a set of rules associated to each application and specified in an XML file called the Deployment Descriptor of the application.

In the context of HTTP-based environments, a HTTP Servlet [3.50] runs within a Web server extended with Servlet Container capabilities. HTTP servlets are derived from the generic Java servlet described previously. All the previous information describing servlets apply also to them. In this type of servlets, the *service* method receives standard HTTP requests, in the form of an *HttpServletRequest* object and dispatches them to the *doXXX* methods defined in this class. Figure 3.9 shows the internal process in a Web server upon reception of a request to a servlet. The first time an HTTP servlet is invoked in an HTTP servlet container, the *init* method of the servlet is invoked, configuring and putting the servlet in service. After that, or for successive invocations of the servlet, the *service* method is invoked.

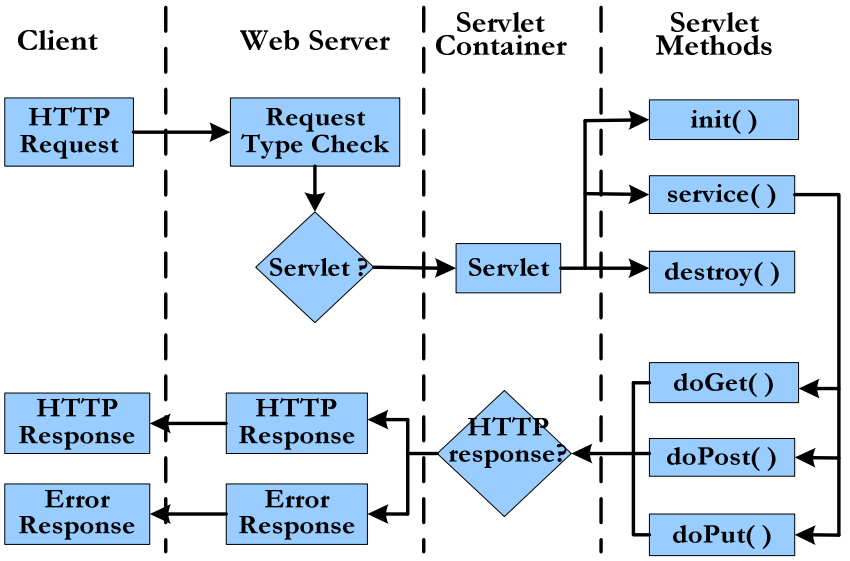



Figure 3.9 Request-Response to/from an HTTP Servlet

The *destroy* method of the servlet is invoked by the container when taken the servlet out of service. The servlet's *service* method receives the standard HTTP requests from the container and dispatches them to the different *doX* methods, according to the type of request received (only *doGet*, *doPost* and *doPut* appear in the figure. The API provides methods to handle Trace, Options Head and Delete HTTP requests too).

### 3.5.2. SIP Servlets API

The SIP Servlet API grew as the Java Specification Request 116 [3.47]. It was started in April 2001 with the aim to define a high-level extension API for SIP Servers, enabling SIP applications to be deployed and managed based on the servlet model. The final release [3.51] was published in February 2003.

Just like HTTP Servlets, SIP Servlets extend the Generic Servlet class in the javax.servlet java packet and so the packages javax.servlet and javx.servlet.sip must be supported by a SIP Servlet container. Due to the differences between SIP and HTTP, in nature and so in applicability, the SIP Servlet API, apart from the common capability of responding to incoming requests, offers extended capabilities like request initiation, request and response reception, request proxying or response sequencing. SIP Servlet applications have to register an interest in the events they should be triggered by. The Service method of the javax.servlet.Servlet interface delivers these events (incoming requests / responses) to the application. The process of a request or a response arriving to a SIP Server in which a SIP Servlet-based application is running appears in Figure 3.10 and described in the following.

When a SIP message arrives at the Servlet Container, this entity checks if the arriving message matches any of the triggering rules for the SIP servlets it hosts. If so, one of the Servlet's methods is invoked. The *init* method is invoked the first time a SIP Servlet is triggered in order to configure it and put it in operation. The servlet's destroy method is invoked by the container when un-deploying a servlet. The *service* method invokes, through the *doRequest* and *doResponse* methods, subsequent methods depending on the characteristics of the arriving message itself. The result of the operations in the final method may be of no (SIP) action at all, of creation of a new SIP request or creation of a SIP response.

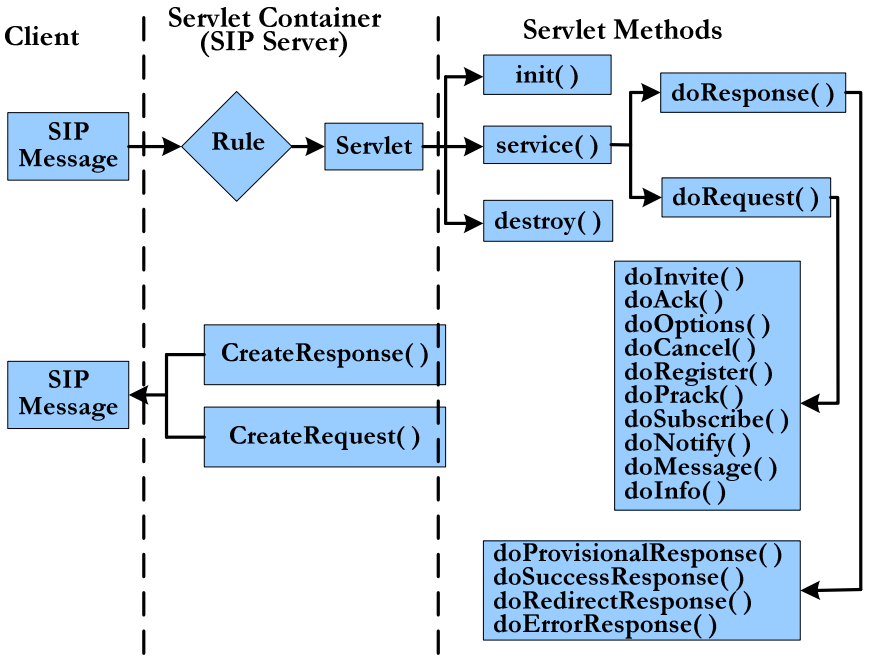



Figure 3.10 SIP Servlet Message Handling Model

### 3.5.3. SIP Servlets API Reference Implementation

With the SIP Servlets API release and as a proof of concept, Dynamicsoft Inc., a major player in the development of the SIP Servlets API, released a SIP Servlets API Reference Implementation (SSARI) as well as a Technology Compatibility Kit (TCK) based on the mentioned reference implementation [3.52]. This TCK contains a SIP Server integrating a SIP Servlet Container that can be used by developers to hosts their applications based on the Reference Implementation of the API.

This Reference Implementation, and the TCK's SIP server, has nevertheless an important shortcoming: it provides support for TCP only as a transport protocol for SIP messages. This implies that for any service based on it, a TCP connection has to be opened and maintained by the entity accessing the server. This may lead to scalability problems when resources are exhausted in the server hosting services based on the SSARI or in any proxy server feeding it. The Reference Implementation of the API as well as the mentioned Technology Compatibility Kit are updated periodically. The current (February 2005) version of the reference implementation is v1.0.8.

# 3.6. GEMINI IP-SCP

The IP-SCP is an application server, which enables delivery of IN-style services to users in a VoIP environment served by an IP-SSP via SIP protocol, according to the architecture depicted in Figure 3.11.

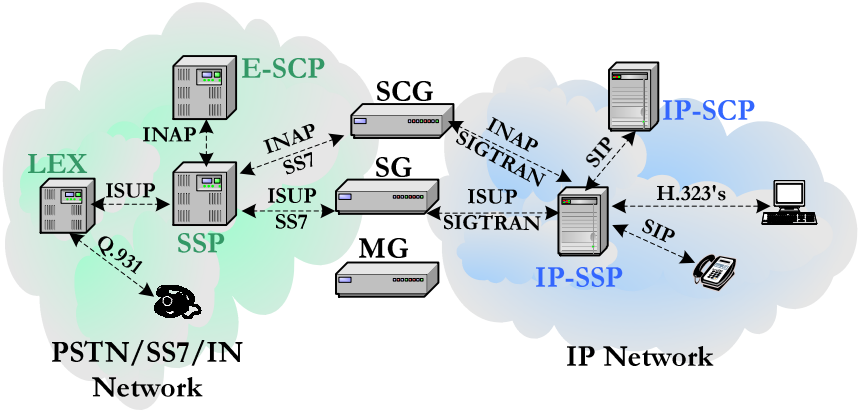



Figure 3.11 GEMINI Architecture

Thanks to the gatewaying and protocol conversion mechanisms detailed in a previous section, PSTN terminals can access also to the services hosted by the IP-SCP.

The behaviour of the services is defined by SIP Servlets in the IP-SCP. The primary function of the IP-SCP is to provide an application execution environment to the services, integrating for that purpose the SIP Server and SIP Servlet Container provided by Dynamicsoft's Reference Implementation and Technology Compatibility Kit for SIP Servlets. The IP-SCP allows for easy access and interfacing for other IP based equipment, integrating a web server, Apache Tomcat's that allows web-based management of the IP-SCP as well as service customization for the subscribers of services. The IP-SCP receives incoming requests for services. Each service is a SIP Servlet, or a set of them. After an initial analysis of the request (validity, target and triggering rules) the IP-SCP identifies the proper service application that should serve it. The IP-SCP then invokes the corresponding SIP Servlet-based application, passing all received parameters to it. The IP-SCP prior, during and at the end of the service execution, may collect additional information from other entities. After executing the service application, the IP-SCP returns the result to the origin of the service request, via the IP-SSP. A relational database, PostgreSQL, is used as service data repository. An schematic architecture for the IP-SCP is presented in Figure 3.12.

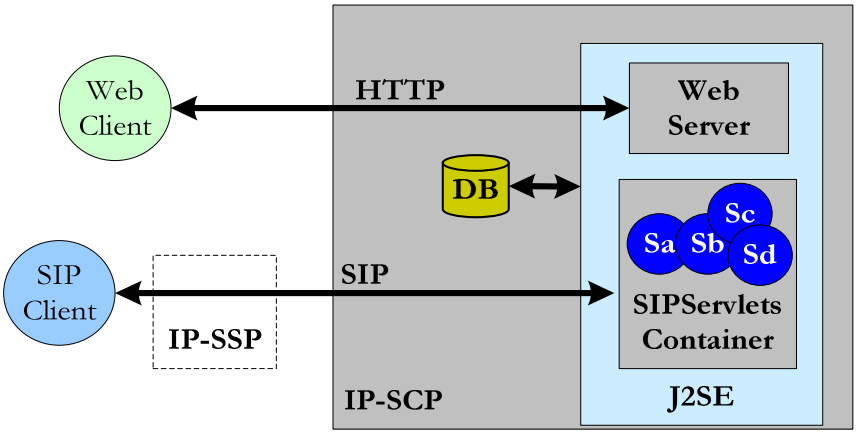



Figure 3.12 IP-SCP schematic architecture

## 3.6.1. GEMINI Services

To test the capabilities of the IP-SCP and the capacity of the GEMINI architecture to host and provide IP-based IN-like services, two different services were designed and implemented by the author to be hosted by the IP-SCP as SIP Servlets: Televoting and Calendar-based Call Forwarding.

### 3.6.1.1. Televoting Service

The Televoting service is an IN service that allows the realization of telephone-based surveys, where the creator of a poll (companies, TV or radio stations) proposes possible responses to specific questions. Each response is assigned a specific number. Callers wanting to vote simply dial the Televoting number for the chosen option in the poll. The result of the survey is determined by the amount of calls to the proposed options. For polls aimed to gather an accurate public opinion, a requirement is that each person votes only once. On the other hand, it might be desirable for the company initiating the poll, i.e. premium rate per Televoting destination and commercial purposes, that the number of votes per voter is not limited.

For GEMINI, a SIP based Televoting service should be hosted in the IP-SCP. The service configuration interface should allow the initial creation and set-up for the poll, i.e. valid period for reception of votes, number and content of options, and voter's behaviour (allowed frequency for vote

repetition). In order to check the status of a poll, a user should be able to access the IP-SCP through a web-page where information about the poll, i.e. contents, validity time, poll options, should be displayed as well as the results so far for the poll if it is active and not finished yet. This interface should update and display poll results in real-time.

As displayed in Figure 3.13, a voter, using a SIP terminal[2], sends its vote towards the poll number. This vote is proxyed by the SIP-Proxy within the IP-SSP towards the Televoting service implemented in the IP-SCP. For simplicity, the IP-SSP has been omitted in the drawing.

The Televoting Service in the IP-SCP is a Java Application based on SIP-Servlets. There is a single application running for all the existing polls. It connects to the proper databases to validate and register the vote within the corresponding poll. The Management Interface towards the service is web-based, allowing the creation of a poll, the deletion of a poll, the access to the existing polls as well as the information about the current status of a poll. The users (voters) have web access to check the status of a poll they may be interested in.

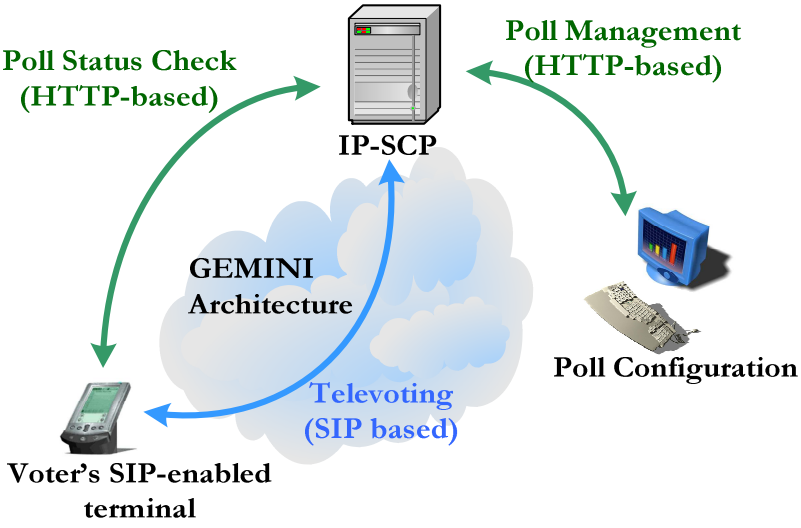



Figure 3.13 Televoting Service

The Televoting service was devised so that the address for each of the poll options follows the following pattern:

OptionId_Pollname_*Poll @ gemini.com.dtu.dk*

[2] Voters in the PSTN side of the architecture are able to access the service also. Details of the mechanisms allowing it are provided at the end of this section.

As displayed in Figure 3.14, when a voter relays its vote, a SIP INVITE message is sent from its terminal towards the IP-SCP through the IP-SSP.

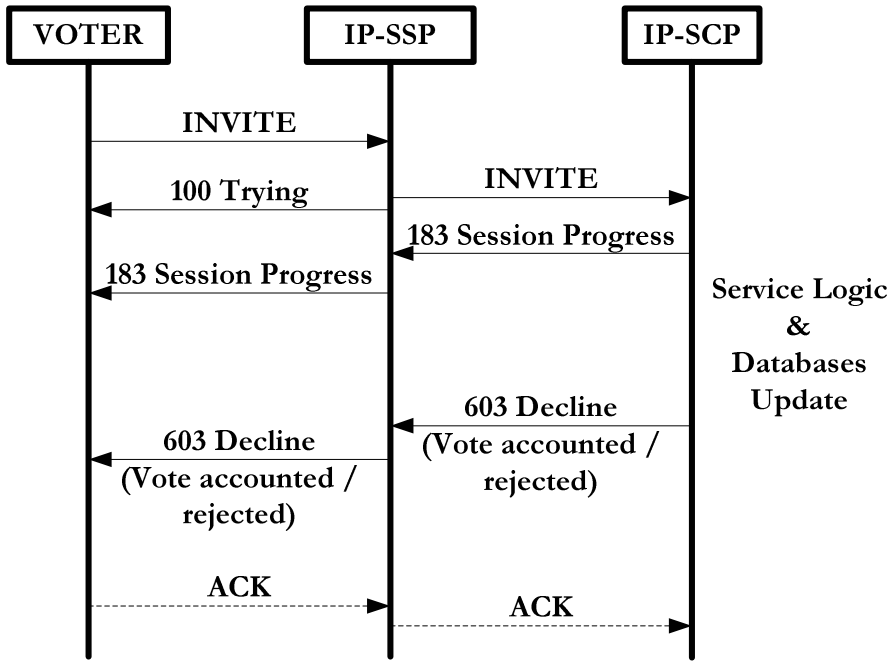



Figure 3.14 Televoting Service SIP Message Flow

The SIP INVITE arrives at the IP-SSP whose internal SIP Proxy recognizes the "poll" pattern within the TO field of the SIP INVITE, based on previous static configuration of the internal SIP proxy of the IP-SSP. Thanks to it, no registration of polls or poll options is needed from IP-SCP to IP-SSP. When the voting message arrives to the IP-SCP, the SIP Servlet Container analyses whether it matches any of the triggering rules for the Servlet based applications it hosts. The SIP Servlet container is configured so that every SIP INVITE message that arrives and contains the pattern *poll* within the TO field will be passed to the SIP Servlets-based Televoting application. The Televoting service SIP Servlet, when receiving the SIP INVITE message replies with a provisional SIP response with code 183 "Session Progress" back to the voter terminal through the IP-SSP so that the terminal knows that its request is being processed. After that, the *Televoting* SIP Servlets-based application parses the URI of the TO field of the incoming INVITE message to extract the Poll Name and the Option the vote is targeted to. Proper mechanisms handle votes to non-existing polls, votes to non-existing options within a poll or votes out of the allowed time-span to vote for a poll, as null votes. A definitive SIP response with code 603 "Decline" and Rea-

son Header, explaining the cause of the rejection (i.e., “Null vote: the poll is not alive”, “Null vote: the poll does not exist”) is sent back to the client through the IP-SSP.

In case the vote is valid, the application will update the proper tables of the DB structure. Once the vote has been properly registered, a SIP definitive response with code 603 “Decline” and Reason Header (i.e.: “Your vote was accounted. Thanks for joining GEMINI´s poll service”) is sent back to the voter through the IP-SSP. Once this 603 message is sent, the application does not care about the compulsory ACK message back from the client, since TCP is used as a transport protocol and there should be no problems on the 603-message transmission-reception.

The choice of the 603 mechanism, instead of a SIP 200 OK message, in order to confirm to the user the proper reception of the vote was done in order to simplify the service operation as well as the messages flow: In case a SIP 200 OK message is used, this would imply the creation of a SIP dialog that should consequently be monitored and closed by means of a SIP BYE message as well as the necessary SIP ACKs during the whole phase. The chosen solution, based on a SIP non-200 OK definitive response is optimum from that point of view: The chosen 603 “Decline” definitive response, allows using it for both rejection and confirmation of votes and so it simplifies the whole Televoting service operation.

For voters in the PSTN side of the GEMINI architecture, an E.164 number is assigned per poll-option and the exchanges configured to route the call signaling towards the signalling gateway (SG). The IP-SSP receives the Initial Address Message (IAM) ISUP message from the SG, over SIGTRAN, and maps it to a SIP INVITE message. The target address of this SIP message is the corresponding to the proper option and poll, thanks to an internal mapping mechanism from E.164 number to SIP URI. This makes necessary to update this numbering information within the IP-SSP whenever a new poll is created or closed. When the 603 response, from the IP-SCP, ending the service is received in the IP-SSP, an ISUP Release (REL) message is generated towards the SG over SIGTRAN. From the SG, the REL ISUP message is transmitted back to the local exchange serving the user.

To test the service, several polls were created with different characteristics (voting repetition not allowed, vote repetition allowed, vote repetition allowed without violation of a minimum time interval) and different test cases were considered (voting to existing poll and existing option, voting to existing poll and options within the allowed voting period, outside it, violating the allowed voting frequency and complying with it). The quali-

tative behaviour for the service tests was satisfactory. Performance details are provided in the following chapter.

After the in-lab, all-IP-based testing, when integration of the IP-SCP with the rest of elements in the architecture was performed, the same tests were carried-out using clients located in the PSTN side of the GEMINI architecture with the same results, except the following drawback: when the signalling relative to the voting call arrives to the signalling gateway, there is no information being given to the "voter" before the call is cleared and so, no information about the acceptance/rejection of the vote is given. This could be solved by means of intelligent peripherals within the PSTN/IN side of the architecture.

The results of the tests and service demonstration were satisfactory. The integration of IP-based technologies and interfaces to create, configurate and follow the poll results by means of web pages were quite convenient. This web interface was based on several HTTP Servlets created by the author. The service logic creation, once the functionality and mechanisms of SIP Servlets comprehended, it was easily performed and debugged.

#### 3.6.1.2. Calendar-based Call Forwarding (CbCF)

Once the basic Televoting Service was satisfactory completed a more "sexy" service was designed. Call Forwarding is a simple IN service that allows forwarding calls to a third destination according to the preferences of the target of the call and an a priori configuration. In traditional IN, call forwarding is categorized as:

- Unconditional, when every call setup request for a target number is forwarded to another destination.
- Conditional, when the forwarding occurs, only if the destination is busy or does not answer to the call setup request.

GEMINI's Call Forwarding is a special case of conditional call forwarding since the forwarding operation and its target are not determined by the status of the subscriber's terminal but by the configuration of its MS Outlook Calendar for the time the call request arrives to the IP-SCP, as drafted by Figure 3.15, and explained in the following.

The subscriber of the service configures it, so that a request to initiate a call is forwarded according to the busy / free information of its personal calendar. This information is formatted as a standard vCalendar file with extension .vbf by MS Outlook's.

When a caller, using a SIP terminal, initiates a call towards the service subscriber, this initial request is proxyed by the SIP-Proxy within the IP-SSP (not included in the drawing) toward the CbCF service implemented

in the IP-SCP. The CbCF in the IP-SCP is a SIP Servlets-based application.

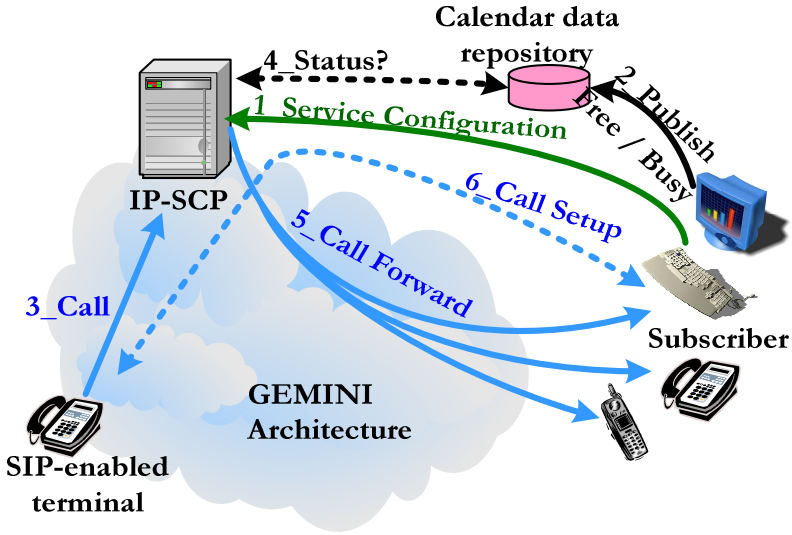



Figure 3.15 Calendar-based Call Forwarding Service

Depending on the status of the subscriber, the CbCF forwards the call according to the subscriber's configuration: if free the call is forwarded to the Internet phone of the subscriber. If busy, the call is forwarded to different terminals or an Email may be sent to the subscriber notifying of the arriving call, according to the subscriber's configuration for the service

The set of SIP messages, the CbCF service is based on, appears in Figure 3.16 for the case of busy state of the subscriber and notification of the call via email.

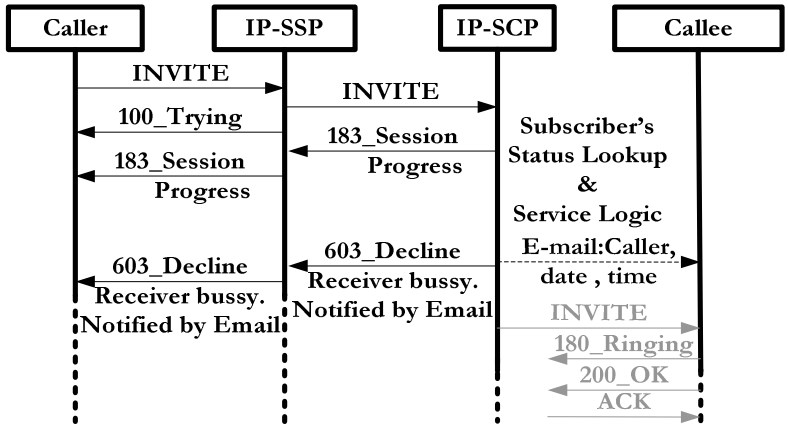



Figure 3.16 CbCF Service Messages Flow

In this case, the call initiator receives a 603_decline SIP response with information about the notification of the call to the callee via e-mail (SMTP).

For the rest of possible cases (free status, or forwarding on busy status) the message flow is the standard for SIP proxying operations from the IP-SCP to the proxying target, as appears in the figure, and a basic SIP setup message exchange follows (only the messages involved in the session setup at the Callee interface are displayed in Figure 3.16, in grey)

It is important to note that this service constitutes also an example of a Virtual Private Network (VPN) service, since the forwarding operation is based on an alias of the final destination of the call.

As in the case of the Televoting service, the CbCF service was primarily tested and debugged in-lab with different software SIP clients. During the project integration phase the service was again tested with “hardware” SIP phones as well as with conventional phone terminals in the PSTN side of the GEMINI architecture. Even an SMS-gateway provided by Telekom Austria was integrated in the architecture so that the service e-mail based warning could be transformed in an SMS to a mobile phone terminal.

These experiences demonstrated the capability of SIP Servlets for creation of powerful SIP-based applications that, integrated within the GEMINI architecture, can be presented as IN services in hybrid telephony networks.

Although the results were satisfactory, considerations on the architecture and service deployment, led to the sophistication of the service deployment mechanisms as presented in the following chapter.

## 3.7. Summary

A comparative description of H.323 and SIP based service architectures has been presented as well as an historical description of initiatives and mechanisms towards interoperation between networks integrating voice service as another data service in an IP environment, and PSTN/SS7/IN in the so-called Hybrid Networks.

The GEMINI network, an example of a hybrid architecture, has been introduced and some of the possibilities for new services have been described along with its underlying technology, i.e. SIP Servlets. SIP Servlet-based services designed, developed and tested by the author within the GEMINI architecture have been described.

# 3.8. Chapter References


[3.1] J.Soler, A.Fosgerau, B.Grabner, "Intelligent Services in Converged Networks-Evolution Steps in the Signalling Arena". 10th International Conference on Telecommunications. Papetee, Tahiti. February 2003. Proceedings book. ISBN: 0-7803-7662-5.

[3.2] J.Soler, A.Fosgerau, M.Stenhuus, "SIP Servlets: an Efficient Scheme for Implementing Advanced Telecommunication Services". 3rd International Conference on Networking. Gosier, Guadeloupe. March 2004. Proceedings book. ISBN: 0-86341-325-0.

[3.3] ITU-T, Recommendation H.323 (11/96), "Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service". November 1996. Superseded.

[3.4] ITU-T, Recommendation H.323 (07/2003), "Packet-based multimedia communications systems". July 2003.

[3.5] H. Shculzrinne et al., IETF RFC 1889, "RTP: A transport Protocol for Real-time Applications". January 1996.

[3.6] ITU-T, Recommendation H.225.0 (07/2003), "Call Signalling protocols and media stream packetization for packet-based multimedia communication systems". July 2003.

[3.7] ITU-T, Recommendation Q.931 (05/98), "ISDN user-network interface layer 3 specification for basic call control". May 1998.

[3.8] ITU-T, Recommendation H.245 (01/2005), "Control Protocol for multimedia communication". January 2005.

[3.9] ITU-T, Recommendation H.235 (08/2003), "Security and Encryption for H-series multimedia terminals". August 2003.

[3.10] ITU-T, Recommendation X.680 (07/2002), "Abstract Syntax Notation One (ASN.1): Specification of basic notation". July 2002.

[3.11] ITU-T, Recommendation H.323 (11/2000), "Packet-based multimedia communication systems". November 2002. Superseded.

[3.12] ITU-T, Recommendation H.450.1 (02/98), "Generic functional protocol for the support of supplementary services in H.323". February 1998.

[3.13] ITU-T, Recommendation H.450.12 (07/2001), "Common Information Additional Network Feature for H.323". July 2001.

[3.14] M.Handley et al., IETF RFC 2543, "SIP: Session Initiation Protocol". March 1999. Obsoleted.

[3.15] J.Rosenberg et al., IETF RFC 3261, “SIP: Session Initiation Protocol”. June 2002.

[3.16] M.Handley, V.Jacobson, IETF 2327, “SDP: Session Description Protocol”. April 1998.

[3.17] 3GPP TS 23.228 v6.8.0 ”IP Multimedia Subsystem (IMS)”, December 2004.

[3.18] H.Schulzrinne, J.Rosemberg, “A comparison of SIP and H.323 for Internet Telephony”.Proceedings of NOOSDAV. July 1998.

[3.19] R.Glitho, “Advanced Service Architectures for Internet Telephony: A critical overview”. IEEE Network Magazine, July / August 2000.

[3.20] J.Glassmann, W.Kellerer, H.Müller, “Service Architectures in H.323 and SIP: A comparison”. IEEE Communications Surveys & Tutorials. 4th Q 2003.

[3.21] Nortel Networks, Protocol Selection Recommendation for UMTS, “A comparison of H.323v4 and SIP”. January 2000.

[3.22] R.Stewart et al., IETF RFC 2960, “Stream Control Transmission Protocol”. October 2000.

[3.23] K.Morneault et al., IETF RFC 3331, “SS7 Message Transfer Part 2 User Adaptation Layer”. September 2002.

[3.24] G.Sidebottom et al., IETF RFC 3332, “SS7 Message Transfer Part 3 User Adaptation Layer”. September 2002.

[3.25] G.Sidebottom et al., IETF RFC 3868, “Signaling Connection Control Part User Adaptation Layer”. October 2004.

[3.26] R.Stewart, Q.Xie, “STCP. A reference guide”, Addison-Wesley. ISBN: 0-201-72186-4.

[3.27] S.Fu, M.Atiquzzaman, “SCTP: State of the art in research, products, and technical challenges”, IEEE Communications Magazine, April 2004.

[3.28] P.Faltstrom, M.Mealling, IETF RFC 3761, “The E.164 to URI Dynamic Delegation Discovery Subsystem Application (ENUM)”, April 2004.

[3.29] ITU-T, Recommendation E.164 (05/97), “The international public telecommunication numbering plan”. May 1997.

[3.30] P.Mockapetris, IETF RFC 1034, “Domain Names: concepts and facilities”, November 1987.

[3.31] T.Berners-Lee et al., IETF RFC 2396, “Uniform Resource Identifiers (URI): Generic Syntax”, August 1998.’

[3.32] G.Camarillo et al., IETF RFC 3398, “ISUP to SIP Mapping”, December 2002.

[3.33] G.Camarillo et al., IETF RFC 3578, “Mapping of ISUP overlap signaling to SIP”, August 2003.

[3.34] A.Johnston et al., IETF RFC 3666, “SIP-PSTN Call Flows”, December 2003.

[3.35] J.Peterson et al., IETF RFC 3842, “Using E.164 numbers with SIP”, June 2004.

[3.36] L.Slutsman et al., IETF RFC 3136, “The SPIRITS Architecture”, June 2001.

[3.37] S.Petrack, L.Conroy, IETF RFC 2848, “The PINT service protocol: Extensions to SIP and SDP for IP access to Telephone Call Services”, June 2000.

[3.38] J.Lennox, H.Schulzrinne, T.F.La Porta, “Implementing Intelligent Network Services with the Session Initiation Protocol”,Technical Report. February 1999.

[3.39] F.Haerens, Internet Draft (draft-haerens-sip-inap-00), “INAP Support of the SIP/SDP Architecture”, October 1999.

[3.40] V.Gurbani, Internet Draft (draft-gurbani-iptel-sip-in-imp-01), “SIP enabled IN services – an implementation report”, November 2000.

[3.41] V.Gurbani, V.Rastogi, Internet Draft (draft-gurbani-iptel-sip-to-in-05), “Accessing IN services from SIP networks”, August 2001.

[3.42] V.Gurbani, F.Haerens, V.Rastogi, Internet Draft (draft-gurbani-sin-02), ”Interworking SIP and Intelligent Network Applications”, June 2002.

[3.43] V.Gurbani, PhD Thesis, “Service Oriented Computing: enabling cross-network services between the Internet and the Telecommunications Network”, December 2004.

[3.44] http://portal.etsi.org/tispan/TISPAN_ToR.asp

[3.45] http://www.3gpp.org/About/about.htm

[3.46] 3GPP TS 23.228 v6.8.0, “IP Multimedia Subsystem (IMS)”, December 2004.

[3.47] http://www.jcp.org/en/jsr/detail?id=116

[3.48] M.Hall, “Core Servlets & Java Server Pages”, Prentice Hall, ISDN: 0-13-0893404

[3.49] http://java.sun.com/products/servlet/2.2/javadoc/javax/servlet/package-summary.html

[3.50] http://java.sun.com/j2ee/1.4/docs/api/javax/servlet/http/package-summary.html

[3.51] A.Kristensen (Dynamicsoft), “SIP Servlet API Specification, version 1.0”, February 2003.

[3.52] http://www.sipservlet.org/

# 4. Network Neutral Service Deployment

This chapter introduces network neutral telephony service invocation. A brief overview of historical trends toward this aim is presented. After that, a discussion on current trends in distributed applications and computing service architectures is provided with the introduction of Web Services and related technologies.

An application of these concepts to service architectures in telecommunication networks is provided and how this was carried out by the author in the GEMINI architecture, with and overlay introduced for service invocation based on Web Services mechanisms. This work was presented in [4.1]. A specific single request / single response protocol, designed by the author, to work as decoupling mechanism between the specific signaling protocol, SIP or INAP, and the signaling protocol independent, web services-based applications hosting the services are presented. A redesign for the GEMINI services was performed based on this mechanism. A performance analysis (service execution times) of the different service implementations is provided.

## 4.1. Protocol Independent Service Architectures

Service architectures and service deployment mechanisms have been presented in the previous chapter. Those architectures and the service logic they support are strongly influenced by the underlying signalling protocols. SIP Servlets and services based on this technology were presented in the case of the GEMINI architecture.

While developing service logic with the underlying protocol mechanisms at hand allows exploiting the most of its capabilities, this has some drawbacks. Firstly, the service developer has to be an expert in the communication protocol(s), its features and operation. Secondly, the application of the service is constrained to the network/s supporting that protocol. This results in service logic duplication/replication in the case of service provision to different networks. [4.2]

In order to address these drawbacks, service architectures have been devised, based on higher level Application Programming Interfaces (APIs) hiding the details of the underlying networks and its protocols. As a result, applications (service logic) based on these APIs can be deployed in any network supporting them and the application developers need lower level of knowledge about the underlying network technology details. This leads to a network convergence in the service layer, and service logic independency from the evolution or the underlying protocols and technologies. Three architectures, with major impact in the industry, are presented in the following subsections.

### 4.1.1. TINA

For almost a decade (1993-2000), the Telecommunications Information Networking Architecture (TINA) Consortium [4.3] worked in the definition of architectural principles for telecommunication infrastructures based on advanced software technology. Based on a business model differentiating the roles of service and connectivity provision, the TINA architecture proposes a separation between service intelligence, generalized call control, connectivity and transport technology. Having the transport technology and the service architecture independent from each other, allowed network neutral service design, implementation and evolution.[4.4]

Based on Common Object Request Broker Architecture (CORBA) [4.16] as distributed software technology, TINA's work was a challenging effort and a seminal work to synergies between software and telecommunication architectures.

### 4.1.2. JAIN

In 1997 a Java based API for programming integrated telephony call control was released as Java Telephony API (JTAPI) [4.5]. Targeting the programming of private branch exchanges (PBXs), JTAPI allows the creation of call control based services for the closed environment the PBX is serving to. In order to extend java based telephony call control to

public networks, the Java community process started in 1998 the specification of a set of APIs resulting in the Java Applications for Integrated Networks (JAIN) architecture specification.[4.6]

The JAIN architecture is divided in three layers as shown in Figure 4.1. The lowest layer is formed by the different JAIN protocol specific APIs that generalize the access to the different stacks for each protocol, making the applications, based on those APIs, independent of any specific vendor's stack. JAIN Call Control (JCC) and Coordination and Transactions (JCAT) APIs provide a common call model for the underlying protocols and a programming interface towards it [4.7][4.8]. While JCC allows applications to initiate and manipulate calls, JCAT allows applications to be invoked before and during the cycle of a call [4.9][4.10]. Java Service Logic Execution environment specifies a service platform for execution of event oriented applications and the way carrier grade telecommunication services can be built, managed and executed on it [4.11].

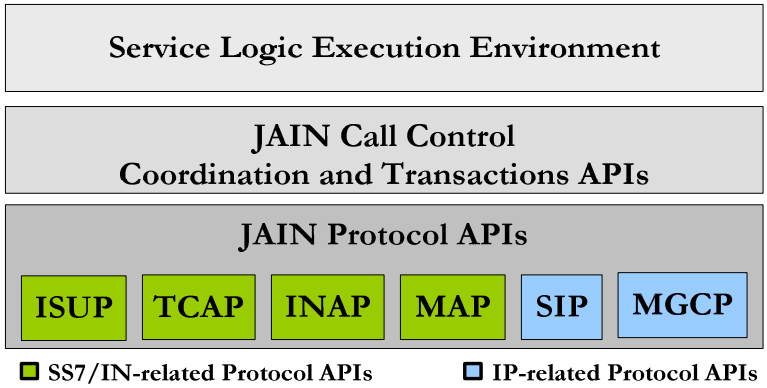



Figure 4.1 JAIN Architecture.

## 4.1.3. PARLAY / OSA

In order to foster competence in the service area, national regulatory bodies pushed dominant telephony operators to allow third party service providers access to its switches [4.12].

With an open access to the interfaces of the switches, the network integrity and security would be compromised. To face this security threat and limit the disclosure of its network interfaces British Telecommunications started, with a group of vendors, the Parlay Group [4.13] to define a secure, open interface to telephony switches [4.12]. The Parlay interface in

a telephony operator's network appears represented conceptually in Figure 4.2.

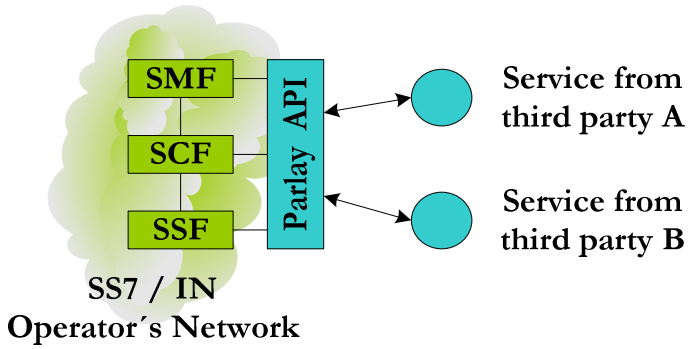



Figure 4.2 Parlay API role.

As represented in the picture, the Parlay interface offers third party providers accessibility to the service switching function as well as the service control and service management.

With a similar aim as Parlay, ETSI and 3GPP started working on the Open Service Architecture (OSA) [4.14] for UMTS networks. Both specifications Parlay and OSA converged from Parlay version 3 and OSA release 5 [4.12].

## 4.2. Web Services and Service Oriented Architectures

Web Services emerged to provide an open alternative to architectures based in proprietary binary protocols supporting remote object models such as CORBA's IIOP and as solution to the problems that networking mechanisms presented to distributed applications based on them [4.15][4.18]. Powerful middleware software technologies, i.e. CORBA [4.16], with proven capabilities in closed, managed environments, probed to fail in the open incipient Internet where firewalls prevent the non authorized data flows to / from different networks [4.17]. This made the configuration of the underlying network infrastructure necessary in order to deploy/access to applications communicating via the Internet Inter-ORB protocol (IIOP), CORBA transactions are based on [4.18], enabling the proper security policies in the affected network firewalls. On the other hand, most of the public and private network infrastructures keep the access to the HTTP protocol ports open in order to allow the access to Web Pages from/to outside the controlled environments. This fact made HTTP

an ideal protocol to carry remote procedure calls (RPC) requests and responses, allowing in such a way the communication between distributed software components along different networks and overcome its security policies.

The issue of how to formalize the messages and the mechanisms for these procedure calls was solved by using the text-based, platform-independent standard for data description Extensible Markup Language (XML) [4.19]. After the initial work of Microsoft and IBM, the Simple Object Access Protocol (SOAP) [4.20] was released and soon gained industry acceptance as middleware mechanism allowing the exchange of messages between software components. SOAP does not specify the use of any underlying protocol as transport mechanism for its XML-formatted text messages, but only a binding for HTTP was defined.

The software components, being part of distributed applications or complete applications in themselves, communicating via web mechanisms (SOAP over HTTP) as middleware, are termed Web Services in the context of this thesis.

These software components would only be reusable if they provided the capability to describe its interface: how to invoke them and what to expect from that invocation, i.e. the messages they accept and the parameters or element types that these messages should contain, both for requests and responses. This functionality is provided by another XML based specification, Web Services Description Language (WSDL) [4.21].

SOAP and WSDL provide the capabilities to access and describe web services. But a need to discover which Web Services were available, its capabilities and location was still missing as a key to reuse those software components to build applications. This functionality is provided by the Universal Description, Discovery and Integration (UDDI) specification [4.22].

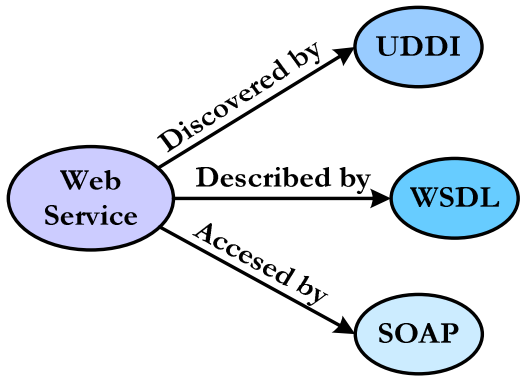



Figure 4.3 Web Services' Technologies

Together SOAP, WSDL and UDDI provide the necessary functionalities to create a Web Services Model were each web service can be found and located, its interface described and its functionality accessed in order to create distributed applications as grouping of individual software components [4.22].

This set of software components, providing independent functionality, which can be combined without constraints to provide extended capabilities or services and thanks to web (HTTP & SOAP) mechanisms acting as middleware, is defined in this thesis as a software service oriented architecture. This architecture provides a horizontal integration layer of vertically different technological components as shown in Figure 4.4.

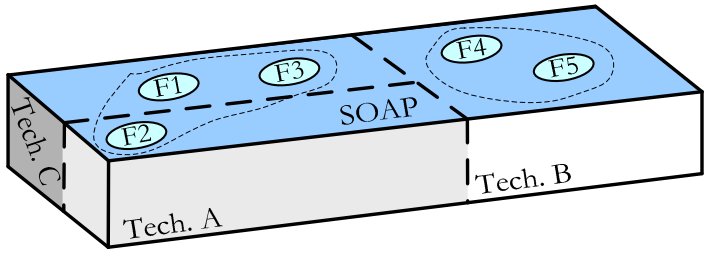



Figure 4.4 Web Services Technologies as Service Integrator

As an example, Figure 4.4 illustrates the case in which a series of units of functionality developed following a Web Services approach may collaborate together to form applications, defined here as the coordination between different units of functionality (Web Services). The applications are represented in the figure by dashed lines. SOAP acts as the common mechanism enabling this interaction and despite the web services may be hosted in technologically different network platforms, as represented by the greyed supporting blocks in the figure.

### 4.2.1. Parlay X

While the Parlay API is based on CORBA as communications middleware, Parlay X API provides a Web Services based interface to a set of the Parlay functionalites provided by an operator's network [4.24]

With Parlay X not only software components or applications can be built on the telephony operator's network capabilities but applications themselves are open to be accessed from those networks through the ParlayX and based on its web services characteristics.

# 4.3. GEMINI architecture revisited

Reviewing the GEMINI architecture that was presented in the previous chapter and displayed again for convenience in Figure 4.5, it is possible to point out a problem.

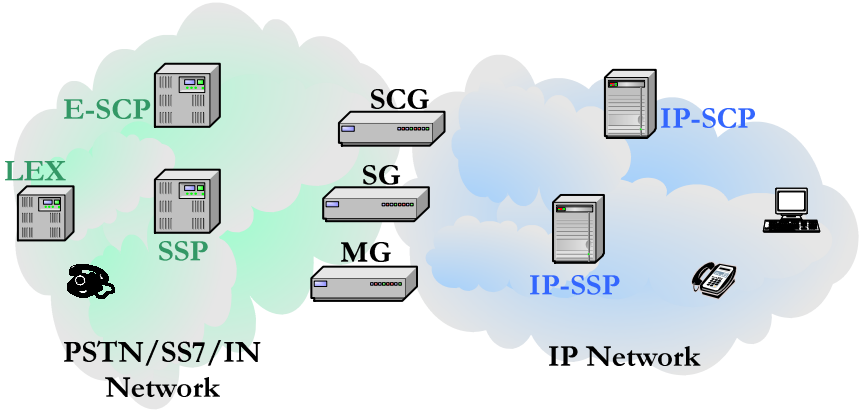



Figure 4.5 GEMINI Architecture

Services offered in the IP side of the architecture are based on SIP Servlets and therefore tightly coupled with the underlying network architecture. As shown in Figure 4.6, if a service is requested by a telephone terminal in the PSTN/SS7 side, it is necessary first a transport mechanism of the SS7 signalling protocol over IP in order to reach the IP-SSP. This is provided through SIGTRAN by the signaling gateway system (SG and SCG).

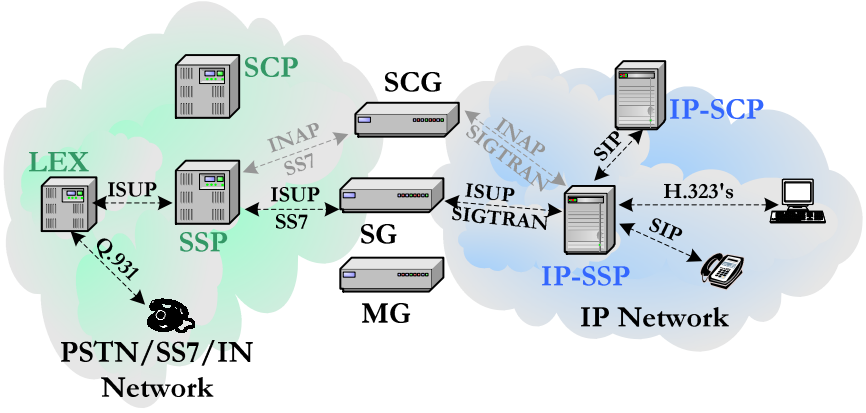



Figure 4.6 GEMINI's initial service invocation mechanism.

Secondly, a protocol conversion from the initial signalling protocol in the requesting network to the SIP protocol, in order to invoke the service, is also needed. This conversion is also necessary in the case of service invocation by an H.323 entity in the IP side of the architecture.

An interesting option, due to performance reasons (IP-SSP offload, SS7's timers) would be to invoke services hosted in the IP-SCP from the PSTN/SS7 side directly. Invoking them from the SS7's switches towards the IP-SCP, in the same way they would access IN services hosted in SCP's in the SS7 architecture, via INAP. Carrying these INAP flows in the IP side of the architecture, would be done over SIGTRAN, as displayed in Figure 4.7.

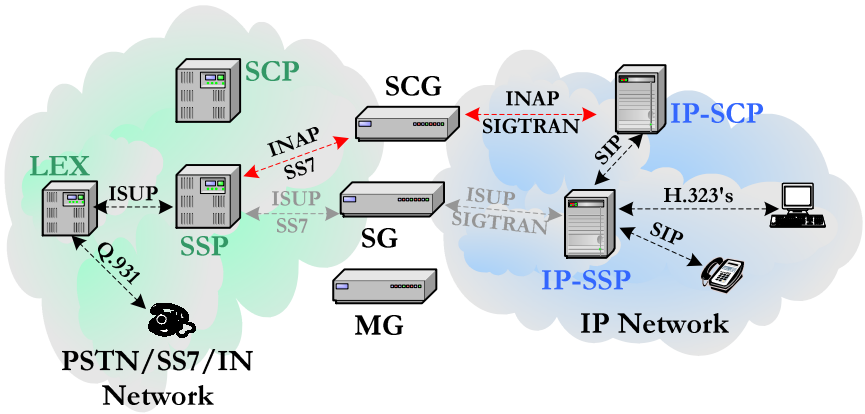



Figure 4.7 GEMINI's alternative service invocation mechanism.

This would require, in the IP-SCP, an INAP/TCAP stack over SIGTRAN and a conversion mechanism from INAP to SIP in order to access the SIP Servlet-based services or a decoupling mechanism from the signalling protocol (SIP, INAP) and the service logic, as shown in Figure 4.8. The later solution was the adopted as described in the following subsections.

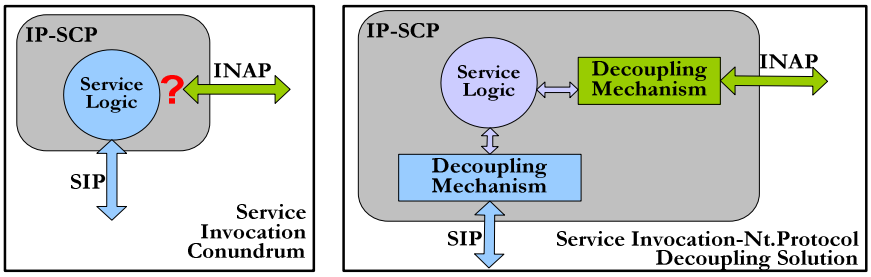



Figure 4.8 Service Invocation problem in the IP-SPC and proposed solution.

### 4.3.1. A Unifying Service Layer.

The solution adopted to solve the "service invocation conundrum" in the Gemini architecture is based on a decoupling mechanism from the network signalling protocol and the service logic invocation.

The solution bases on the service oriented architecture concepts presented previously, with web services mechanisms as enabling technology. In order to achieve protocol independent services, they were built as web services by the author, with a single request / single response protocol as service invocation method and content of the web service's SOAP messages. This service invocation method was termed GEMINI IN (GIN) protocol.

As a result, and considering servlets, java applications with communication capabilities through a request/response protocol, applications built with GIN messages as the contents of SOAP-RPC exchanges were termed GIN Servlets. Therefore GIN Servlets provide an open interface, described by WSDL, towards the service logic, via GIN. This makes the service logic accessible from any environment, provided the necessary HTTP connectivity, allowing an application developer, without knowledge of the details of the underlying signalling network protocols (SIP/INAP) to program applications / services.

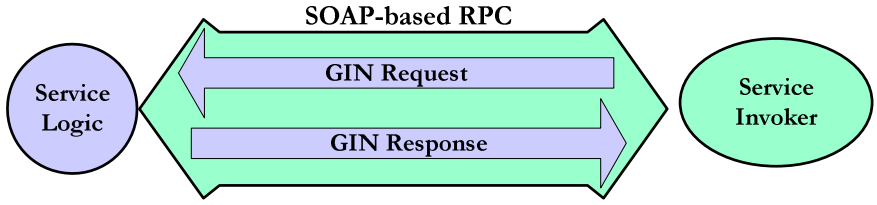



Figure 4.9 GIN-based service invocation

#### 4.3.1.1. GIN. Defining the message exchange.

In order to define the messages that the applications implementing the services should use in their invocation and response, a trade-off was considered: while creating a multipurpose set of messages for use of any kind of present and future service / application would be desirable, this was not achievable. Reality forced us to limit the scope, since scheduling constraints needed to be considered for developing and testing within the GEMINI project activities.

Therefore the set of services upon consideration was limited to the kind of services implemented already in the GEMINI architecture: Televoting

and Calendar Based Call Forwarding services were already implemented, and in order to demonstrate the concept they would be developed as GIN Servlets.

So a set of messages suitable for both services and services sharing its characteristics was the target.

In order to find a common denominator for these two services, its service behaviour was comparatively analysed. As shown in Figure 4.10 and described in the previous chapter, the SIP-servlet based televoting service was based on a 3 basic messages mechanism, basically a request (SIP's INVITE request) a reception confirmation (183 Session Progress SIP response) and a service completion message with the result of the service execution, vote accepted / vote rejected & reason (603 Decline SIP response).

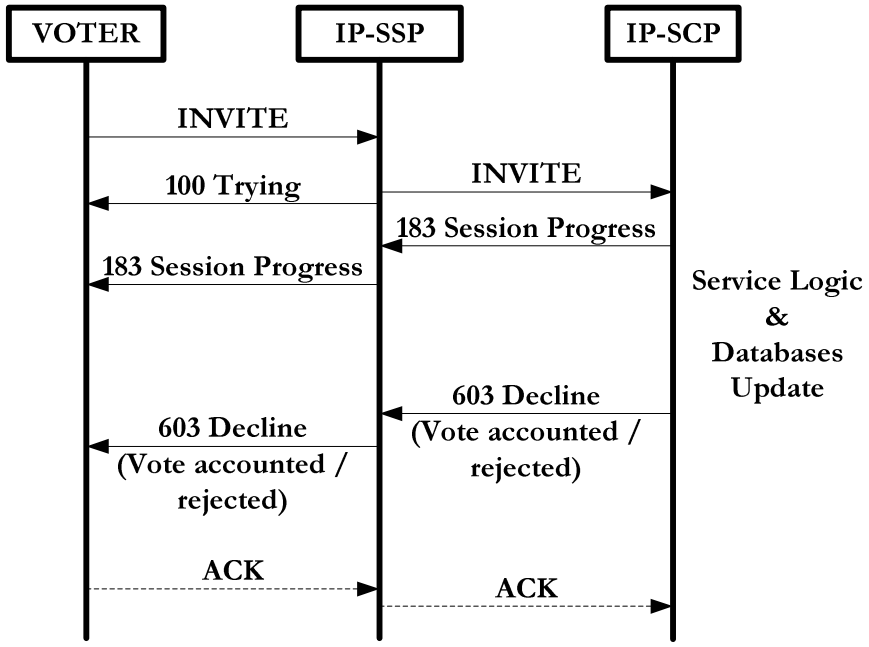



Figure 4.10 Initial design for the Sip-based Televoting service

On the other hand, the Calendar based Call Forwarding mechanism involved a proxying operation from the initial session start-up request towards the proper target, according to the service user options and as shown in Figure 4.11.

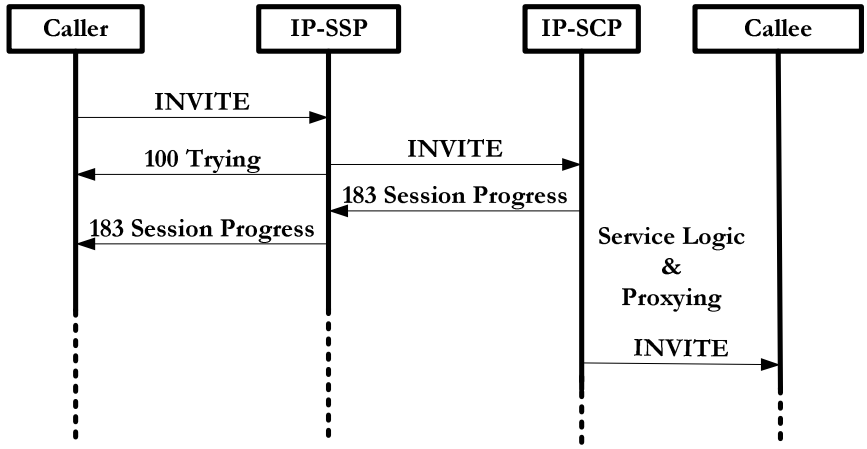



Figure 4.11 Calendar Based Call Forwarding SIP session start-up.

The Televoting service was then modified, by the author, to behave in a different way. Due to this modification and as shown in Figure 4.12, after the outcome of the service process (valid vote accounting or rejection of invalid), the initial SIP request is proxyed to a second SIP Servlet, termed Aux Servlet in the figure and different in each case (vote accepted / vote rejected). This auxiliar Sip Servlet transmits to the service client the proper final message, terminating the service.

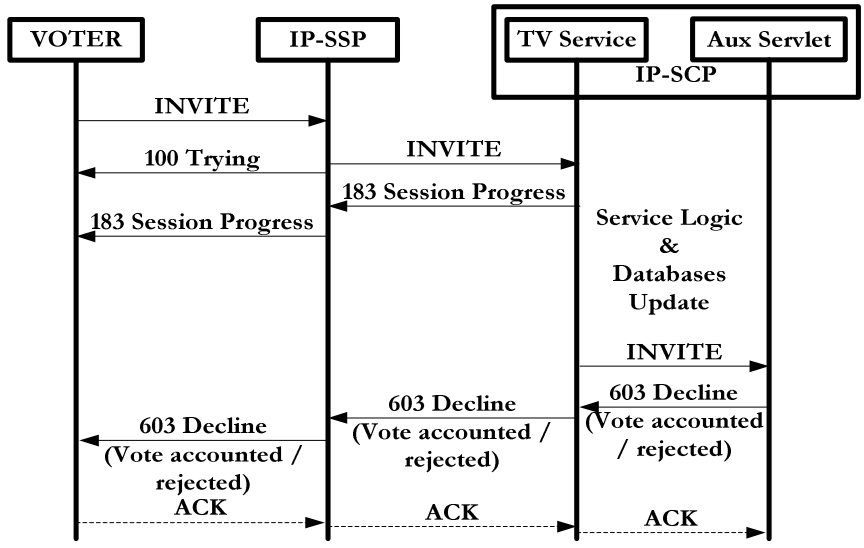



Figure 4.12 SIP-Servlet based Televoting Service with auxiliary SIP Servlet

With this modification, the service behaviour as seen from outside the IP-SCP remains identical but, the internal service mechanism was now based on a proxying mechanism as a way of terminating the service: If the vote was valid and accepted the initial INVITE message is proxyed by the Televoting service to an auxiliary SIP Servlet responsible of replying positive answers (603 Decline SIP message with Vote Accounted response) while if the vote is not valid the initial INVITE is proxyed to a different auxiliary SIP Servlet which replies with negative answers (603 Decline SIP message with Vote Rejected and cause response).

It is clear by comparing Figure 4.11 and Figure 4.12 that despite the service logic for these services is still different, the signalling procedures are now common, based on a proxying operation after the main service logic process.

Based on these common features, the method to decouple the service logic from the signalling protocol was designed by the author, targeting all those services that could be built over a proxying operation as a mechanism to terminate the service. Termed GEMINI IN (GIN) messages, the service invocation mechanism was designed with the following considerations:

- GIN based transactions are stateless: all the information necessary for the service execution should be available upon service invocation. GIN mechanism does not support services with additional request for information from the service logic. From an IN / INAP point of view this means that GIN does not support “Collect Additional Information” flows and only a single detection point per service is allowed.
- All destinations are equal: no differentiation between intelligent peripherals and an endpoint are made from the point of view of the service logic.

The input to the application implementing a GIN-based service is a GIN Request. It consists of the “number” of the caller and the called “number”.

A GIN Request is defined in WSDL as follows (simplified):

```
<complexType name="GINRequest">
  <sequence>
      <element name="called" type="xsd:string"/>
       <element name="caller"  type="xsd:string"/>
  </sequence>
</complexType>
```

As it appears in the WSDL definition of the GIN request, these "numbers" are mere character strings. It is assumed that these strings represent SIP uniform resource identifiers (URIs), where PSTN E164 numbers have been converted using ENUM. The entity creating the GIN request is responsible of performing this conversion and validating the SIP URIs.

The service application then responds with a GIN response described in WSDL as follows (simplified):

```
<complexType name="GINResponse">
  <sequence>
    <element name="n" type="xsd:int" />
    <element name="response"
     type="impl:ArrayOf_tns1_GINResponseElement" />
  </sequence>
</complexType>
```

This WSDL description simply states that the GIN response is a compound of an integer and an array of "Gin Response Elements" were each response element is defined as a set of 4 strings, each of them representing different target "numbers" and as shown below (simplified).

```
<complexType name="GINResponseElement">
  <sequence>
    <element name="id" type="xsd:string" />
    <element name="normalTerm" type="xsd:string" />
    <element name="bussy" type="xsd:string" />
    <element name="noAnswer" type="xsd:string" />
  </sequence>
</complexType>
```

The *id* element is the first number the reply is targeted to, being the other a list of alternative destinations, for the following different possible outcomes of the call:

- Terminates normally.
- Destination is busy.
- Destination does not answer.

Therefore, the application returns a "graph" of destinations, which are called sequentially, depending on the outcome of a call. This allows support for two-step services like "conditional call forward" or "play announcement and then forward", which would not otherwise be possible, as the GIN transactions are stateless. The processing ends, when a node without further links to the graph has been reached. The principle is illustrated in Figure 4.13.

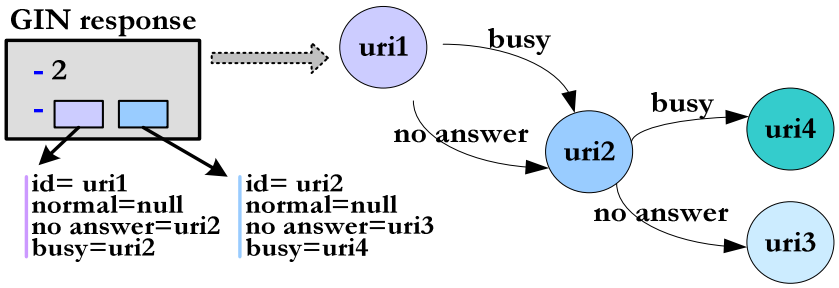



Figure 4.13 GIN response and destination graph example.

The figure shows the example of a GIN response containing 2 response elements, as indicated in its initial parameter. In the event that the target for the first response element id, uri1, is busy or does not answer, the next node in the graph is the second element in the response, whose id is uri2. If the call to this element's id, uri2, is not answered, the next node in the graph to be tried is uri3, while if uri2 were busy, it would be uri4 the next node to follow in the graph.

#### 4.3.1.2. GIN-Mapping.

The messaging mechanism to interact with GIN based service logic has just been presented. A single GIN request is replied with a single GIN response, containing a graph of destinations as a final outcome for the service. It is the responsibility of the requestor to handle the information within the GIN response.

In the case of the GEMINI architecture, GIN services were developed in the IP-SCP. These services can be accessed directly via GIN by applications behaving like web service clients. In order to allow other entities in the GEMINI architecture to access these services, i.e. SIP clients or PSTN terminals, a mapping from the underlying SIP or INAP messages to GIN request / responses was devised.

In Figure 4.14 the access to a GIN based service via INAP flows is illustrated. As shown there, a mapping application from INAP to GIN deals with the specific INAP details and is in charge of the iterative forwarding process that the GIN response enables.

When a preset detection point in the basic call state model (BCSM) is triggered (i.e. service number), the SSP sends an InitialDP INAP message to the INAP/GIN mapping application. There, the information to construct the GIN request is extracted from the INAP message and the GIN service is invoked. The GIN service returns a GIN response that is passed back to the INAP/GIN mapping application. This entity starts execution

of the iteration through the destinations graph contained in the GIN response.

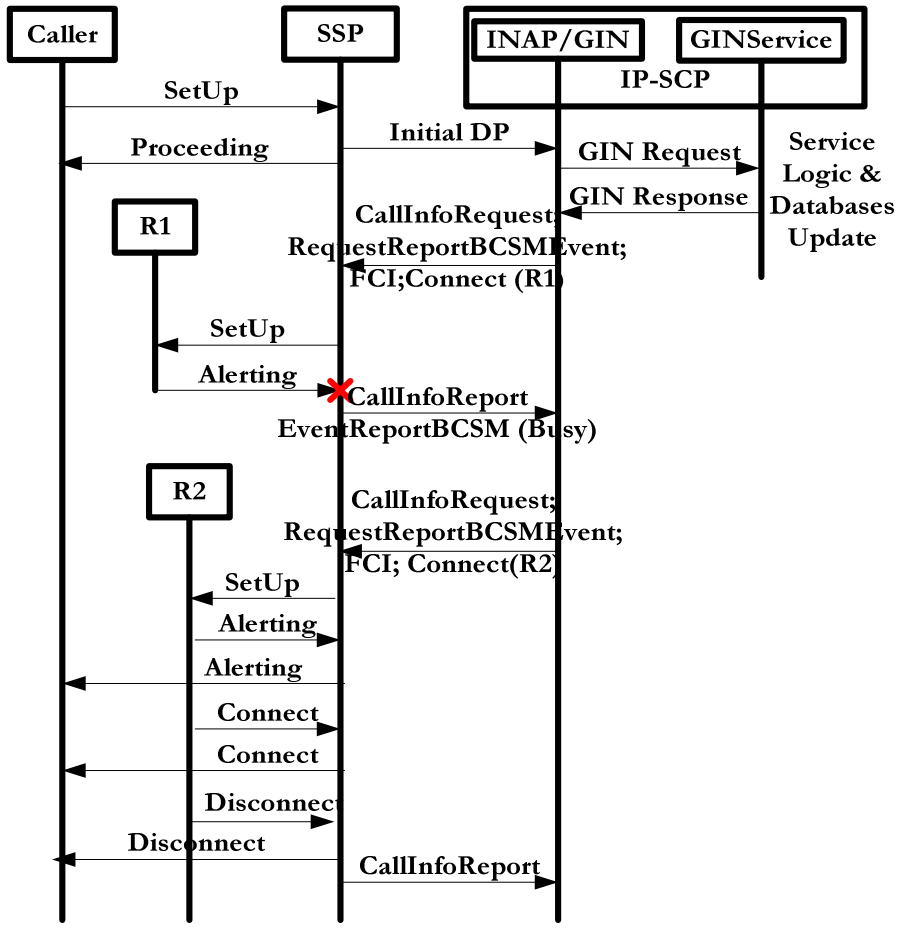



Figure 4.14 Accessing a GIN-based service via INAP.

As an example, Figure 4.14 shows the case in which the first option for a destination to be contacted at the end of a GIN service execution is Receiver 1 (R1). The INAP/GIN mapping application requests to the SSP to initiate a connection towards R1 as well as to report occurrences of a Busy or No Answer event in the BCSM of this call. The example displays the case in which the receiver R1 is busy at the connection set-up trial. The INAP/GIN mapping application in the IP-SCP would be reported accordingly by INAP flows and it would iterate through the graph of destinations to the next target in the graph of destinations received in the GIN response. In the example, Receiver 2 (R2) is so targeted and the

connection is completed this time. At the end of the call the INAP/GIN application would receive the notification of call completion.

It has to be noted that the destination endpoints for the service termination are not considered to be Intelligent Peripherals (IP), implementation of an IN's Special Resource Function, but as mere endpoints. This allows the use of the same INAP flows for the services. Targeting a SRF entity would imply the use of a ConnectResource INAP message instead of a simple Connect message. This would make the INAP/GIN implementation service dependent, since differentiation of the nature of the endpoints ending a service (simple endpoint vs SRF entity) would be required[3].

In order to validate the GIN mechanism, the mapping was implemented and tested for the SIP protocol. The application performing the SIP/GIN mapping mechanism is both a SIP-Servlet and a web service client. As a SIPServlet, the SIP/GIN Mapping entity receives from the SIPServlet container those SIP requests targeting GIN services, according to the rules specified in its deployment descriptor. When a SIP request arrives, the SIP/GIN mapping entity converts the parameters from the initial INVITE into a GIN request and invokes the proper GIN service through the SOAP based RPC mechanism with the GIN request as content. Upon reception of the response of the invoked SOAP RPC, with a GIN response as content, the SIP/GIN mapping entity proxies the initial SIP INVITE message iteratively, according to the destination graph contained in the GIN response and the outcome of each of these trials.

Considering the stateless nature of SIP Servlets, implementing the SIP/GIN Mapping entity based on them was somehow "tricky". The solution by the author, though simple, achieved its task: The SIP/GIN mapping entity was implemented as a SIP Servlet. An instance of a HashMap class was added as attribute of this SIP Servlet. This attribute holds instances of a class termed *State* represented in Figure 4.15.

The attribute *index* of the *State* instance represents which element in the GIN response was used in the last processing of the call graph for the GIN response contained in the other attribute.

---

[3] The mapping mechanism towards INAP was only studied theoretically due to the final GEMINI architecture characteristics: there was no testing possibilities since the INAP capabilities were never integrated in the IP side of the GEMINI architecture.

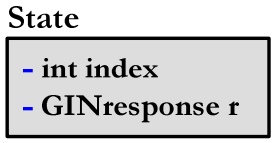



Figure 4.15 State class representation

The key for the *State* instances in the HashMap of the SIP Servlet is the caller's identification. So, when a GIN response is received from the GIN service, a proxying operation is performed and a *State* instance created with the *index* value of 1.

If the signalling due to the SIP proxying operation is forced to pass by the mapping SIP Servlet it is possible to use the *doErrorResponse* method of the SIP Servlet to monitor for 408 (Request Timeout), 480 (Temporarily Unavailable) and 486 (Busy here). If any of these responses is received, the HashMap is accessed for the State instance associated to that call (caller), assumed that a caller is not involved simultaneously in more than a call. According to the *GINResponse*'s graph within the *State* instance and the *index* attribute, indicating the position within the graph it is possible to proceed with the next forwarding operation for the graph as explained previously and to update the S*tate*'s instance values or purge this instance from the HashMap attribute in the event of successful proxying operation.

In the case of successful call completion, the *State* instance is deleted from the HashMap attribute of the mapping servlet.

### 4.3.2. GIN Services

The services developed for demonstration on SIPServlets were ported to GIN Servlets. As explained in previous sections, these GIN Servlets are web services with GIN requests and responses as the content of the SOAP messages inherent to web services.

The Televoting service was simply ported to a web service were the contents of the service request and responses are GIN requests and responses. As displayed in Figure 4.16, when a service invocation (vote) arrives at the IP-SCP, the SIP Servlets-based SIP/GIN Mapping application, parses the URI of the TO field of the incoming INVITE message to determine the name of the GIN-based service to be called and its parameters: the Poll Name and Option the vote is targeted to in the case of the Televoting Service. With those parameters, it constructs the GIN Request and sends

it to the GIN Servlets-based Televoting application, running as a "web-service".

Once this application receives a request, it performs the same set of operations as the SIP Servlets-based Televoting application: validation of vote and votes count update in DB. As result, a GIN Response is sent back to the SIP/GIN Mapping application. In this case the contents of the GIN Response are all null, except the *id* element.

When the GIN Mapping application receives back this response, it proxies the initial INVITE message to the location contained in the *id* element of the GIN response. Two different SIP Servlet based applications were created, *OkRespons*e and *ErrResponse*. They act as "Intelligent Peripherals" in the service and upon receiving the SIP INVITE message proxyed by the GIN Mapping Servlet, they send back to the voter the 603 SIP message with the proper message (Vote accounted / Vote not valid and reason).

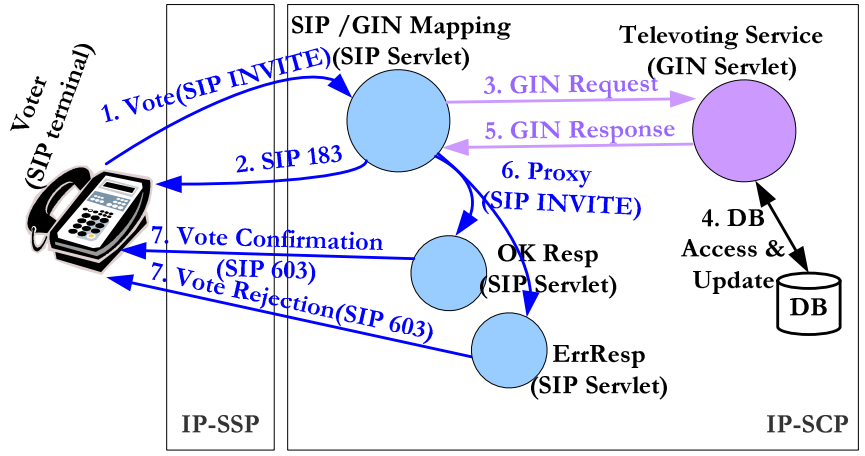



Figure 4.16 GIN-based Televoting Service.

The GIN based Televoting service is a mere Web Service. It can be then invoked by any web service client, implementing its interface (GIN request / response) and by means of SOAP mechanisms. This interface is completely described for the service by means of WSDL. The WSDL description for the GIN-based Televoting service is publicly accessible in the IP-SCP.

In a similar way, the Calendar Based Call Forwarding Service was ported to a GIN servlet. As displayed in Figure 4.17, when the IP-SCP receives the initial INVITE SIP message, the SIP Servlets Container recognises a pattern in the URI of the *To* field of the SIP INVITE message ("*CCFgin*") and passes the request to the *SIP/GIN* mapping application.

After sending the 183 “Session Progress” SIP message, the SIP Servlets-based *SIP/GIN Mapping* application, parses the URI of the *To* field of the incoming INVITE message to determine the name of the GIN-based service to be called and its parameters: the subscriber’s username for this service. With this parameter it constructs a GIN Request, as described, and sends it to the GIN Servlets-based Calendar Based Call Forwarding application, running as a “web-service” in the IP-SCP. Once this application receives the GIN request, it checks from the IP-SCP’s DB the subscriber’s configuration of the service and retrieves her/his MS Outlook Calendar busy/free status from the location specified by the subscriber. If the subscriber status is free a forwarding operation to the Internet Phone of the subscriber must be performed. If it is busy, the call should be forwarded according to the choice of the subscriber during the configuration of the service:

- Forward to an Internet phone (i.e. computer).
- Forward to a Mobile Phone or PDA.

A third option, implying no forwarding at all, but a notification of the call via e-mail in case of busy state of the subscriber has been added to introduce an original element that differentiates the service from a classic call forwarding service. If this is the case, an e-mail is sent to the subscriber.

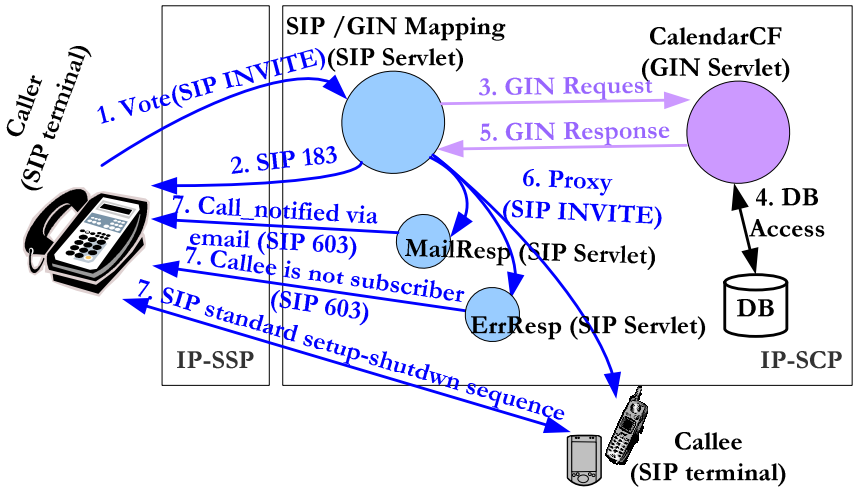



Figure 4.17 GIN-based calendar Call Forwarding service.

When the *SIP/GIN Mapping* application receives back the *GIN response*, it proxies the initial INVITE message to the location contained in the *id* element of the GIN response. In the case of busy status of this endpoint and mail subscriber’s choice as well as in the case of non subscribed target, two different SIP Servlet-based applications were created, *ErrorRe-*

*sponse* and *MailResponse*, that act as "Intelligent Peripherals" in the service. Upon receiving the SIP INVITE proxyed by the *SIP/GIN Mapping Servlet*, the MailResponse SIP Servlet creates and sends an e-mail to the subscriber with a notification of the incoming call, date, time and caller id. Both MailResponse and ErrorResponse send back to the caller a 603-Decline SIP message with the proper message ending the session setup (Mail sent to the busy receiver/ receiver unknown).

### 4.3.3. IP-SCP testing and trials.

The IP-SCP so built was finally integrated and tested within GEMINI's architecture. In order to evaluate, qualitatively and quantitatively, the IP based services, different tests were carried out. Figure 4.18 displays the different messages towards, within and from the IP-SCP during the invocation of a service. It is assumed that the service is closed by a SIP 603 response, what is congruent with the Televoting service in all its cases and the Calendar-based Call Forwarding service when the subscribers would like to receive call notification by E-mail when busy. The upper part of the figure displays the messages for the invocation and execution of SIP Servlets-based service while the lower part displays those regarding the GIN Servlets-based implementation. While both implementations are similar regarding its functionality and the final outcome for the user, its response time and so the performance as sensed by the user is different.

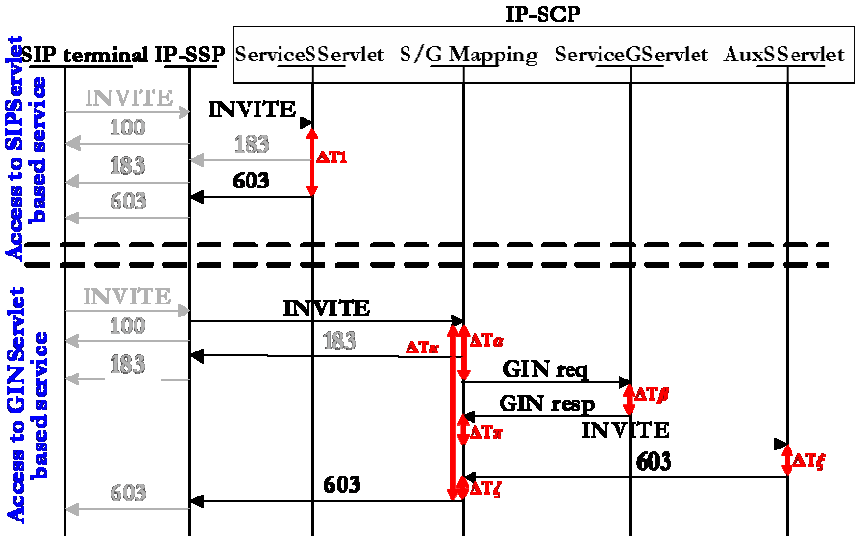



Figure 4.18 Temporal characterization of the service implementations.

Considering neglectable the transmission delays (the integration tests and setup were carried out in a local setup at Telecom Austria premises), the service time, measured as the time interval between the arrival of an initial SIP message at the IP-SCP, invoking the service, and the transmission of the corresponding SIP response finishing the service, for each of the cases can be described as:

- $\Delta T1$
- $\Delta T\kappa = \Delta T\alpha + \Delta T\beta + \Delta T\pi + \Delta T\xi + \Delta T\zeta + \Delta\rho$

ΔT1 is the service time for the SIP Servlet based service, measured as the interval between the arrival of the triggering SIP INVITE message to the IP-SCP and the transmission of the associated SIP 603 message.

ΔTκ is the service time for the GIN Servlet based service, measured as the interval between the arrival of the triggering SIP INVITE message to the IP-SCP and the transmission of the closing SIP 603 message. It depends on the SIP/GIN Mapping SIP Servlet execution time (ΔTα), the service's GIN Servlet execution time (ΔTβ), the operation of the SIP/GIN SIP servlet interpreting the GIN response and performing the SIP proxying operation towards the auxiliary SIP Servlet (ΔTπ), the execution of the auxiliary SIP Servlet (ΔTξ) and finally the signalling through the IP-SCP towards the initiator (ΔTζ). The factor ΔTρ accounts for delays introduced in the setup of the TCP connections internal to the IP-SCP.

These values were measured as displayed in Figure 4.19 and detailed in the following. (The factor ΔTρ was not registered since it does not provide direct information related to the service execution, though its components can be deduced from the figure as the interval between ΔTα and ΔTβ and the one between ΔTπ and ΔTζ).

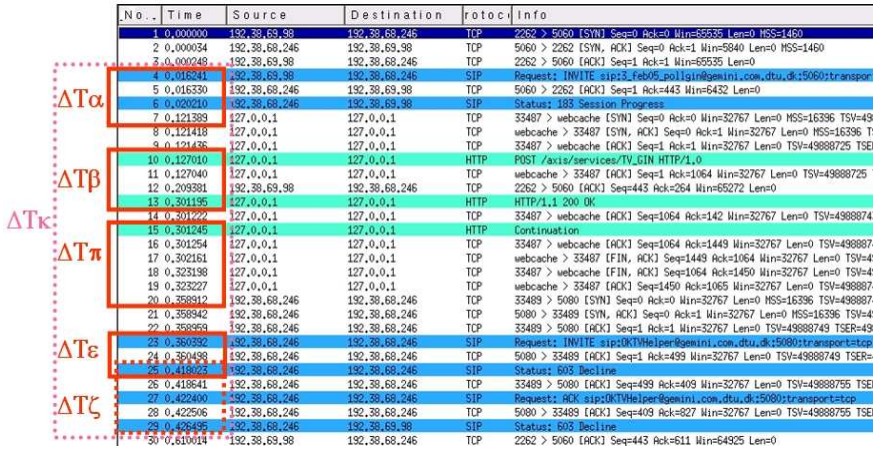



Figure 4.19 Meassuring the service time components

- $\Delta T\alpha$, is measured as the interval between the arrival of the initial SIP INVITE message and the start of the TCP connection supporting the HTTP message that carries the SOAP message with the GIN request. It accounts for the temporal cost of the processing of the arriving SIP message into the SIP Servlet's container (rule matching), triggering the proper SIP Servlet and its execution, included the creation and initiating the transmission process of the GIN request.
- $\Delta T\beta$ is measured as the interval between the transmission of the HTTP message carrying the SOAP message with the GIN request, and the reception of the initial part of the HTTP response. This part is empty (the GIN request is not contained in it) and that is why has not been considered for the calculation of the following factor, $\Delta T\pi$. This factor accounts for the reception of the HTTP message with SOAP content with a GIN request, processing its contents and the GIN based service logic execution with the creation and transmission of the GIN response from the HTTP server and SOAP engine.
- $\Delta T\pi$ is measured as the interval between the arrival of the HTTP response with the SOAP message containing the GIN response and the start of the TCP connection that will carry the proxyed SIP INVITE message. This factor accounts for the processing of the SOAP message contents back into the SIP/GIN mapping SIP Servlet until the proxying operation finishing the service.
- $\Delta T\xi$ is measured as the time between the proxying of the SIP INVITE message and the reception of the final SIP 603 response from the auxiliary servlet. This interval accounts also for the processing time of the SIP message within the servlet container (rule matching and Servlet triggering).
- $\Delta T\zeta$ is measured as the interval between the arrival of the final SIP 603 response from the auxiliary SIP Servlet until the transmission of this response out of the IP-SCP.

The following table describes the times measured for these values and for each of the services tested (no SIP Servlet based CbCF service implementation was finally integrated in the IP-SCP). A discussion of these results follows.

| SERVICE | SERVICE TIME | CONTRIBUTORS | | | | |
|---|---|---|---|---|---|---|
| **TV_SIP** | $\overline{\Delta T1} = 172{,}57$ | | | | | |
| | | $\overline{\Delta T\alpha}$ | $\overline{\Delta T\beta}$ | $\overline{\Delta T\pi}$ | $\overline{\Delta T\xi}$ | $\overline{\Delta T\zeta}$ |
| **TV_GIN** | $\overline{\Delta T\kappa} = 412{,}92$ | 95,23 (23%) | 177,11 (43%) | 58,27 (14%) | 47,38 (11%) | 8,47 (2%) |
| **CbCF_GIN (proxying)** | $\overline{\Delta T\kappa}^{*} = 261{,}72$ | 94,85 (36%) | 100,81 (38%) | 53,26 (20%) | - | - |
| **CbCF_GIN (e-mail)** | $\overline{\Delta T\kappa} = 393{,}35$ | 96,29 (24%) | 162,35 (41%) | 53,41 (13%) | 50,27 (12%) | 6,23 (1,5%) |

Table 4-1 Measured Service Time and contributors (in ms.)[4].

As displayed by these results, the response time of the SIP Servlets based Televoting service, is faster than the GIN Servlets-based implementation as expected, due to the higher complexity of the overall process of mapping to/from GIN and the use of the auxiliary SIP Servlets. As a result, the SIP Servlet-based implementation is almost 2,5 times faster than the GIN-based implementation.

In the case of the GIN-based service implementation, the "heavier" factor is the related to the execution of the GIN Servlet (ΔTβ), and comparable to the total service time of the SIP Servlet-based implementation (ΔT1) for the same service (TV_GIN vs. TV_SIP, in the table).

If we compare the GIN-based Televoting service implementation (TV_GIN in the table) with the GIN-based CbCF, in the case that the option when a call to a busy subscriber is to proxy the call to another address (CbCF_GIN_proxying, in the table), the factor ΔTβ, the GIN logic execution time, is considerably lower in this last service implementation, 43% lower. This highlights the temporal cost of the multiple accesses to the database in the TV_GIN implementation, in order to validate the vote

[4] The aggregation of the contributing times for the GIN based services does not match the total service time. This is because the factor ΔTβ only considers the interval between the transmission of the HTTP message containing the SOAP messages with the GIN requests/responses as body, but not the delay induced by the establishment of the underlying TCP connections. This was done so that ΔTβ represents exclusively the GIN-Servlet execution time. The same applies to the factor ΔTξ and the contribution of the Auxiliary Sip Servlet. The missing intervals form the factor ΔTρ, introduced previously.

and to register it accordingly, compared with the single access to the database and accessing and parsing the busy-free file for the CbCF_GIN service. For this service case, the factors $\Delta T\xi$ and $\Delta T\zeta$ are meaningless, since the proxying operation from the mapping servlet targets a receiver outside the IP-SCP and not an auxiliary SIP servlet as in the other cases. As a result the total service time $\Delta T\kappa^*$ is measured as the interval between the arrival of the initial SIP INVITE and the transmission of the proxied SIP INVITE out of the IP-SCP (what would be the starting of the $\Delta T\xi$ interval).

Comparing the two different options of the GIN-based CbCF service implementation, i.e. proxying the arriving message to an alternative destination vs. notification of call via e-mail when receiver is busy, the main difference is in the temporal cost of the main service logic execution ($\Delta T\beta$). The creation and transmission of the e-mail message is the only difference among them.

Regarding the rest of the parameters, it is significative the cost of the $\Delta T\alpha$, accounting not only for the SIP/GIN mapping SIP-servlet execution contribution (mainly the creation and transmission of the GIN request) but also the processing time of the SIP message within the SIP Servlet container.

Also the difference between the values for $\Delta T\pi$ and $\Delta T\xi$ highlights the cost of processing the GIN response and proxying a SIP message ($\Delta T\pi$) versus processing the SIP message in the SIP Servlet container and proxying it ($\Delta T\xi$).

Introducing time measurement in the service code, allowed estimating the contributtion of the Sip Servlet containers to the execution times. The execution time of the TV_SIP implementation, as measured from the SIP Servlet implementing the service logic was of 34 ms. The execution time of the SIP/GIN mapping SIP servlet, from arrival of SIP INVITE until final proxying of the INVITE (termination of the GIN-based service) was of 192 ms. That is, the effect of the IP-SCP elements supporting the service (network interface, OS, SIP Servlet container processing) means 80% of the service time. This difference increases in the case of the GIN-based service logic execution time, since the processing of the HTTP server and the SOAP engine allowing the hosting of web services (Apache Axis), has to be considered besides the one of the SIP server and SIP Servlet container.

The previous detailed measures were taken with a call intensity of 12 calls per hour. This intensity was increased progressively and no significant differences in the total service time were registered until arriving to an intensity of 60 calls/hour. When the call intensity increased above

those limits, unstable behaviour of the service occurred. The problem was detected in the final proxying operation in the SIP/GIN mapping servlet: Due to loop detection problems within the SIP Servlet container, the auxiliary SIP Servlets were hosted in a separate SIP Servlet container within the IP-SCP. The instability was detected in the TCP connections between both containers when performing the proxying operation. Despite this significative low, maximum accepted load, below carrier-grade-service parameters and since the GEMINI demonstration had a qualitative and not quantitative aim, the issue was not further investigated.

The results of the GEMINI integration setup for the final demonstration proved that the GIN solution, as a decoupling mechanism between protocol-dependent and service dependent issues was a valid one.

The design of the GIN messages, targeting a specific set of services, limits its applicability as a general service logic access mechanism. Nevertheless, similar procedures and messages can be designed per set of services, according to its common characteristics and generalizing the GIN procedure. The recently standardized mechanisms of Parlay X [4.24], based also in Web Services, provide the way to setup calls in a Parlay and Parlay X enabled network and to act according to the outcome of the call setup request, allowing the creation of services similar to those targeted by GIN. Nevertheless, the use of two different API's (Parlay X Call and Parlay X Third Party Call) is required with its different invocation procedures. A very similar approach to the decoupling mechanism devised for the GEMINI architecture, but using a Parlay X web services interface from SIP Servlets towards web services based service logic, is presented as part of the Open IMS Playground of the Fraunhoffer Institute [4.25][4.26] .

## 4.4. Summary

A brief overview of software approaches towards telecommunication service architectures has been provided in this chapter. The middleware capabilities of web services technologies, i.e. SOAP, WSDL and UDDI provide a framework to create service oriented architectures, where applications or services are accessed and cooperate based on them.

The case of the GEMINI experience, where the author used Web Services to decouple service logic from protocol specific operations and so from network specificities has been presented. GIN, a generic purpose mechanism for a constrained set of services has been developed in order to interface with the service logic and as a general mechanism for network independent service access for the considered set of services.

# 4.5. Chapter References


[4.1] J.Soler, A.Fosgerau, “Web Services as enablers of Converged Telephony Services”. 7th International Symposium on Communications Interworking. Ottawa, Canada. December 2004. Proceedings CD.

[4.2] S.Moyer, A.Umar, “The impact of network convergence on Telecommunications Software”, IEEE Communications Magazine, January 2001.

[4.3] http://www.tinac.com/

[4.4] M.Mampaey, A.Couturier, “Using TINA Concepts for IN Evolution”, IEEE Communications Magazine. June 2000.

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

[4.6] http://java.sun.com/products/jain/index.jsp

[4.7] http://www.jcp.org/en/jsr/detail?id=122

[4.8] http://www.jcp.org/en/jsr/detail?id=21

[4.9] Keijar et alt., “JAIN: A New Approach to Services in Communications Networks”. IEEE Communications Magazine. January 2000.

[4.10] Jain et alt., “Java Call Control, Coordination, and Transactions”. IEEE Communications Magazine. January 2000.

[4.11] http://www.jcp.org/en/jsr/detail?id=22

[4.12] Zuidweg, “Next Generation Intelligent Networks”. Artech House 2002. ISBN:1580532632.

[4.13] http://www.parlay.org/about/index.asp

[4.14] http://www.3gpp.org/TB/CN/CN5/CN5.htm

[4.15] K.Mockford, “Web Services architecture”, BT Technology Journal, January 2004.

[4.16] Object Management Group, “Common Object Request Broker Architecture: Core Specification. V3.0.3”.March 2004.

[4.17] H.Abie, “CORBA Firewall Security: Increasing the Security of CORBA Applications”, Telektronikk 2000.

[4.18] Object management Group, “CORBA Firewall Traversal Specification”, January 2003.

[4.19] World Wide Web Consortium (W3C), “Extensible Markup Language (XML) 1.0”, 3rd Edition, February 2004.

[4.20] W3C, “SOAP version1.2 Part 1: Messaging Framework”, June 2003.

[4.21] W3C, “Web Service Description Language (WSDL) 1.1”, March 2001.

[4.22] http://www.oasis-open.org/committees/uddi-spec/doc/contribs.htm# uddiv1

[4.23] T.Erl, “Service Oriented Architecture”, Prentice Hall, 2004. ISBN: 0-13-142898-5

[4.24] The Parlay Group, “Parlay X Web Services Specification, v 1.0.1”, June 2004.

[4.25] T.Magedanz, D.Witaszek, K.Knuettel, “The IMS Playground @ FOKUS – An open testbed for next generation network multimedia services”, International Conference on Testbeds and Research Infraestructures for the Development of Networks and Communities”, Proceedings Book. ISBN: 0-7695-2219-x. February 2005.

[4.26] http://www.fokus.gmd.de/bereichsseiten/testbeds/ims_playground/playground/playground.php?lang=def

# 5. Data Centric Architectures and Models

This chapter provides a different point of view to service architectures. While the previous chapters have discussed different mechanisms for service invocation, this one deals with data, service and user profiles, in current and future architectures. A brief introduction to the data storage mechanism and its evolution in IT environments is used as a base for presenting data storage-based problems in current telecommunication networks. The IST project FlexiNET is presented along with its architectural proposal. A generic data model, created by the author, to support user and service data centralization based on the FlexiNET architecture is introduced as well as an specialization of the model covering some of the needs of UMTS networks. This work was presented in [5.7].

## 5.1. Data storage in IT environments

As described in [5.1], the evolution of data handling and storage in IT environments has suffered a dramatic shift. Enterprise networks built over a client-server model, grew as interconnections of local area networks where desktop computers with relatively low processing and storage capacities acted as clients of application servers with the proper processing capabilities and local storage. When resources allowed it, different applications were located in different servers, each with its own local storage, creating in such a way functionality domains and data domains within them as shown in Figure 5.1.

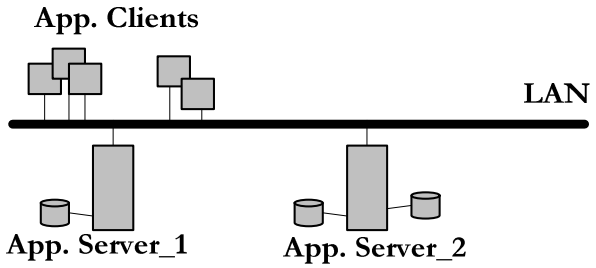



Figure 5.1 Application servers with attached storage.

Within each of these application servers, the data is stored as shown in Figure 5.2. In the storage media, data is made persistent as physical signals, magnetic or optical, which are presented by the disks, hosting the storage media, as numbered blocks of data. The storage capacity of a disk is fixed. Disks can be logically combined in volumes or virtual disks. In order to use volume capacities, file systems provide another level of abstraction allowing the arrangement of the blocks as ordered sequences of variable size, presented in a hierarchical structure of named objects: the files.

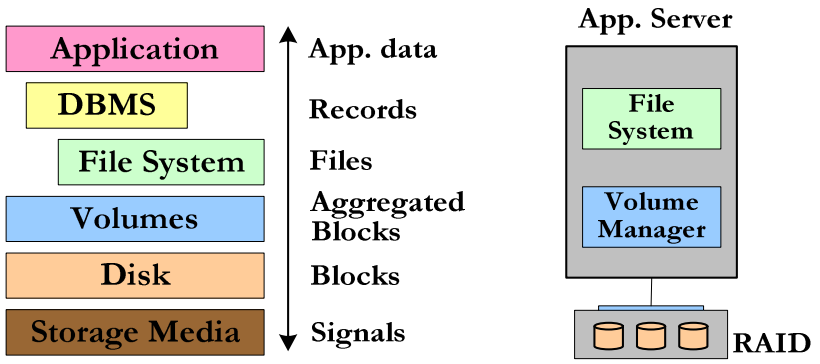



Figure 5.2 Storage in a server [5.1].

Database management systems provide the way of storing application data in an application-independent way. In the case of relational database systems, application data is kept as records within tables in the database. As shown in Figure 5.2 different disks can be combined in redundant arrays of independent disks (RAIDs) and its capacity managed by a volume manager.

The need to share data between different applications and servers led to the generalization in the use of data bases and the standardization of the mechanisms to request and serve data, i.e. Structured Query Language

(SQL) for relational databases. This fostered the separation between application servers and the data they hosted, and the introduction of dedicated database servers with attached storage in the IT infrastructure as shown in Figure 5.3.

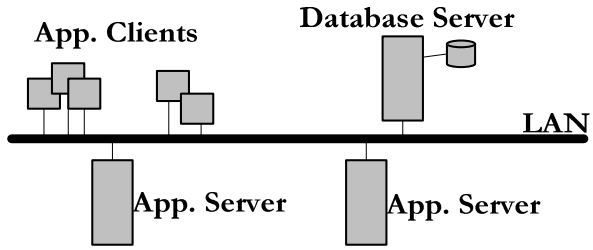



Figure 5.3 Centralising storage in database servers

Concentrating data in single points along the same network allows eliminating the need for data replication between different data storages in different application servers. It also allows eliminating the need for communication between application servers to access their local data and synchronising its contents since all the data is now available to all application servers and data redundancy is no longer necessary. Nevertheless this concentration of data in dedicated servers within the same infrastructure still maintains inefficiency problems when bandwidth resources are exhausted by intensive storage operations affecting applications intercommunication. To overcome these inefficiencies, an independent storage-dedicated network (SN) was incorporated, so that storage operations do not affect the overall network efficiency as shown in Figure 5.4.

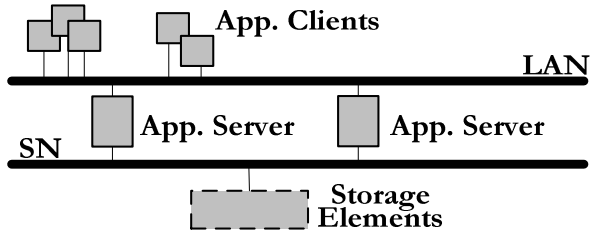



Figure 5.4 Dedicated network for storage operations.

Two different strategies may be found in current deployments of storage networks. In the first one, termed Network Attached Storage (NAS), the application servers behave as file system clients of the file systems of storage serves. They access these storage servers, termed NAS devices, through file access protocols such as NFS [5.2] or CIFS [5.3] via conven-

tional access infrastructures, i.e. Ethernet. The storage capacity is local to the NAS devices as shown in Figure 5.5.

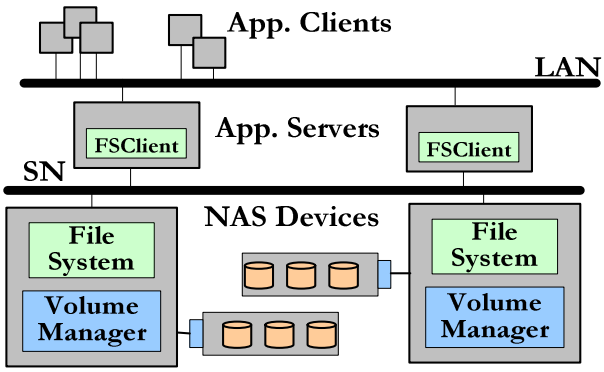



Figure 5.5 Network Attached Storage [5.1]

The second strategy termed Storage Area Network (SAN), is based on a dedicated network infrastructure for storage where the data is managed in terms of volumes and accessed via a block data access protocol such as SCSI [5.4] thanks to a high capacity and high efficiency network infrastructure such as Fiber Channel [5.5][5.6]. As shown in Figure 5.6, in this case the storage capacity of the overall system can be increased by increasing the storage capacity of the SAN and no investments in local configurations are needed.

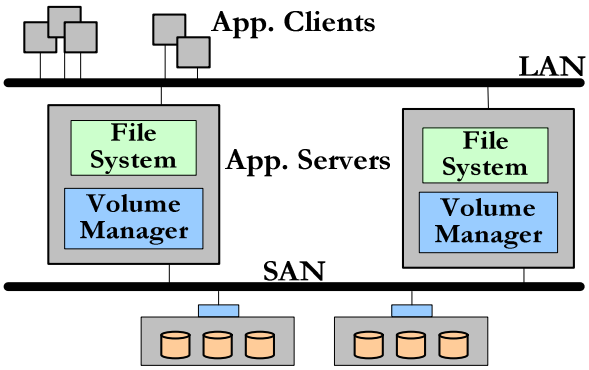



Figure 5.6 Storage Area network [5.1]

The use of storage resources, can be dynamically altered according to the punctual needs of certain applications, providing greater flexibility. Data intensive transfers from application servers to data repositories do not affect the efficiency of the non storage related operations. On the contrary, this efficiency is improved since more transactions can be sup-

ported by the released resources. Special data intensive transfers, i.e. buck-ups or mirroring-related operations are improved and made independently of the state of the network. The SAN becomes a global repository of data, accessible to all the servers connected to it avoiding data replication needs and the related data synchronization [5.1][5.6].

Making an analogy between telecommunication networks and early enterprise networks and application servers it is possible to describe the same problems, regarding data storage [5.7]. Each telecommunication network type has its own representation of operational data and some of these data are replicated in multiple nodes within a single network. As a result synchronization between these pieces of data is necessary to ensure consistency and signalling traffic within the network must be devoted to perform the data updates. Also information about users and services is network proprietary and can not be reused from network to network making the provision of crosservices between different networks cumbersome when not impossible. Homogenization of this persistent information of telecommunication networks would allow its reuse from network to network and interoperability based on it. Centralization of this information in an overlay infrastructure to the existing one would eliminate the need for replication of data and the use of the signaling infrastructure for the related flows. Service logic centralization needs, led to the Intelligent Network model for service provision in telephony networks. An overlay infrastructure centralizing operation, user and service data, would provide the framework for future service provision and interworking between technologically different networks. This idea is developed in the following section.

# 5.2. Converging Networks in the data plane

## 5.2.1. FlexiNET Project

The IST project FlexiNET, started in Spring 2004 with the Research Center COM (DK) as a partner of a consortium formed by Alcatel SEL (D), IBM (CH), Hitachi (FR), Vodafone (D), University of Patras (EL), Teletel (D) and T-plus (D). The aim of the project is to provide a complementary network architecture allowing service access decentralization and independence in the use and storage of user, service and control data from the transport network.

To achieve this purpose, network operation information must be moved from the nodes where it is used to a common data repository in order to achieve consistency and improve network efficiency by reducing the associated signalling within the network. This implies the definition of a generic and unique data interface to this data repository, so that any network can use it as storage and broker of operation and service data, turning it into a universal data centric, interoperability enabler. While this interface will solve accessibility issues to the repository, true interoperability will not be attainable without a common semantic framework for the data contents of the repository, so that common operations and procedures can be applied to the by-nature heterogeneous data from different networks. To achieve this semantic interoperability a generic model for the network information to be accessed through the previous interface and stored in the data repository is proposed, as well as the extension of the model to cover the specific needs of different networks. The result of these processes is a universal network, aggregation of multiple heterogeneous networks, operating user and service information independently of the network it belongs to or is requested from [5.8].

## 5.2.2. FlexiNET Architecture

Figure 5.7 displays the initial architectural proposal in FlexiNET. As shown, this architecture is based on a single data repository, accessed by a generic interface.

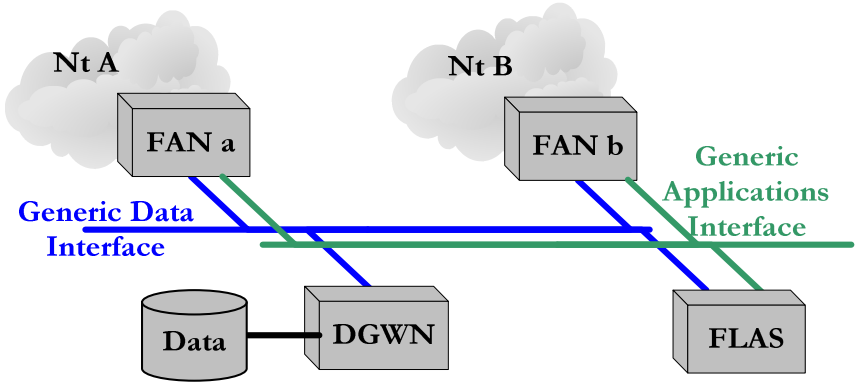



Figure 5.7 FlexiNET initial architecture proposal

A central entity, the FlexiNET Data Gateway Node (DGWN), behaves as data provider for the rest of elements in the architecture retrieving data from the repository and storing data in it. Other elements in the architecture access the DGWN by means of a web services based interface, the

so-called Generic Data Interface (GDI). Basing the interface in Web Services responds to the need for an open interface available to any network attaching to the architecture. The DGWN may be located via UDDI and the interface described in terms of WSDL.

Different networks attach to this architecture via dedicated nodes, FlexiNET Access Nodes, FANs, which provide the signalling mapping between their respective networks and the GDI. Different FANs will exist in the architecture. In Figure 5.7, web services-based interfaces are described in terms of communication busses to represent the difference between interfaces.

A second web services-based interface exists in the architecture. Termed Generic Applications Interface, it allows accessing functionality located in specialized elements, FlexiNET Application Servers (FLASs), from the different Access Nodes (FANs). These FLASs host specialised functionalities, such as authorization, since the DGWN provides no complex service logic but just mere atomic operations over the storage system: storing data or retrieving data. The data repository is based on a storage area network (SAN) providing storage support to a database. The chosen database management system for the architecture is based on an object oriented database according to the Object Data Management Group standards (ODMG) [5.9]. This will make the handling of data in the DGWN, in terms of objects, independent of its persistency status and without the need for mappings between the objects, its attributes and its tabular persistent representation in the case of a relational database had been chosen. The Java for Data Objects (JDO) API [5.10], an evolution of the initial Java binding by the ODMG, hiding the details of the object description data language (ODL) and object query language (OQL) defined by the ODMG and used to define elements and access them in object oriented databases, will be used to access the database from the DGWN. This ultimately makes the design and implementation of the DGWN independent of the chosen database, constrained only by the chosen JDO implementation.

### 5.2.3. A Generic Data Model

Without a generalised and homogeneous treatment of data, independent of the nature or network origin, the FlexiNET proposal would provide a useless architectural framework for network interoperation based on the data plane. To avoid that, a generic model for data and operations affecting those data should be established.

This section presents an initial generic network data model, targeting basic and common network operations in telecommunication networks. Its generality, being its main strength can be claimed also as a drawback since applicability to specific network types requires its extension by specializing to the data needs of that kind of network. A specialization of this generic model, in the case of UMTS networks and a limited subset of operations, is provided in the next section as an example.

In order to create the data model, Unified Modelling Language (UML) [5.11] was chosen, facilitating the development process when using an automatic code generation tool from the model. The chosen tool was the free version of the Omondo EclipseUML [5.12] plug-in for the Eclipse IDE [5.13].

The description of the model herein provided follows a progressive process, describing not only the model elements, its components and relations but the reasons for them and as they appear in the complete model.

The reader is referred to the UML basic diagrams in Appendix 4 and the basic UML notation definitions provided there, in order to follow the textual description of the model.

The objectives to be fulfilled by this generic model are the following:

- To provide a comprehensive mechanism to describe the relations between users and telecommunication networks, understanding by network the set of physical elements they are formed by, the communication services they provide and the mechanisms involved in doing so.
- To enable the definition of a set of generic operations within the network, based on the previous description and defined relations. This set will initially comprehend the following operations: user subscription to network and / or network services, registration of user in the network, access to basic services, use of other networks and billing for the use of the network / services.

User information and the related operations are one of the drivers for the model as explained previously. As displayed in Figure A4.1 (Appendix 4), a user is defined as an entity characterised by its user identification and its personal information. The user identification is a key parameter being unique. Since different networks may attach to the FlexiNET architecture, this parameter will be a compound of the network identifier of the network, the user is subscribed to and its unique user identification within that network. The personal information element (attribute) of the user entity (class) has been defined as an Object type. This will enable to define a new Personal Information class with elements such as name, ad-

dress, bank account information, personal agenda and remainders. Since this is not fundamental information for the operations the model is targeting, this class has not been included in the model, and this can be modified if necessary to include such information mutatis mutandis. A user keeps simultaneous references to two different Network objects, the one he is subscribed to, his “home network” and the one he is using in a particular moment, “the current network”. When the user is using the network he belongs to, his “home network”, the two attributes refer to the same Network object. As described in the UML diagram, a user may have multiple different subscriptions to services in different networks. These service subscription objects, instances of ServiceSubscription class are grouped within an User object in a Map type attribute termed “subscribedServices” where they are accessible using the pair networkId and sserviceId of the service as a key. These service subscription entities, instances of the ServiceSubscription class, have two components. The first one is a reference to the Service the subscription refers to, and the second one is the service configuration of this user for that service. This Configuration entity is service dependent and as such has not been defined. The ServiceSubscription class keeps a reference to the user, so that all users subscribed to a certain service can be reached from the respective subscriptions. The User instances have an attribute termed billingInfo where the user activity is logged. This information can be used to bill the User accordingly by means of a Bill method in the User class that calculates the accumulated expense of the user and for instance sends the bill to the users e-mail address. A user, regardless of its subscription to a certain network can use different networks as well. There may be technological information needs specific to those networks and with persistency requirements. In order to cope with these needs an attribute termed ntDependentData will hold the different subsets of information as a HashMap, allowing keeping information of the user in different networks. In the same way, the user can have different terminals, each with different configuration needs. To cope with that, the attribute terminalConfigs has been added as a map of TerminalConfig elements, each of one keeping configuration details of a different terminal that the User may use and with the brand and model pair as key in the map object. Despite the fact that the user can have multiple terminals, only one can be used at a time and associated as such to a User in the FlexiNET model. As it appears in the model, every user has associated none or one instance of a Terminal class at a certain time, regardless of the number of configurations for different terminals it might have.

Navigating in the model from the User instances, it is possible to arrive to the network the user is subscribed to and the one is using at a certain

time. The instances of the Network class are identified by a unique identifier. Also differentiating attributes of a Network instance are its name and its type, i.e.: UMTS or WLAN. The set of subscribers and users are separated in two attributes, of Map type. While the userId is used as key parameter in the subscribers map, in the users map the access key is the pair formed by the user identification and the network identifier of the network the user is subscribed to. Another attribute of the Network class is the useAgreements map, detailing those networks whose subscribers can use the services provided by the Network instance and at which rates. The key to access the map is the network identifier of the network the agreement refers to. The set of services offered by a network are referenced by the map attribute offeredServices in the Network class instances. A service is identified by its unique identifier. Other attributes provide a description of the service. The attribute called subscriptions is a map type element with references to all the subscriptions related to the service, making the subscribers of the service accessible from them. The subscriberId is the key to access the subscriptions map attribute. Another pair of map attributes, serverCodeVersions and terminalCodeVersions contains references to the different software versions of the service both for the servers hosting them as well as for the clients accesing them from the users terminals. The pair formed by the software platform name and the version of the code is the key for accesing the ServiceServerCode instances in the serverCodeVersions map while the pair formed by the terminal type and the code version of the service is the key for accessing the terminalCodeVersions map of ServiceTerminalCode instances. Different attributes form the service code instances both for servers and for terminals, being one of them a reference to a service configuration, the deffaultConfig attribute. The association between the Service class and the Service Code classes, for server and terminals, is navigable in both directions. Finally, another attribute of the Service class is the billingInfo, instance of the servBillingInfo class, where information about the rate policy for the service is provided.

Going back to the Network class contents, another attribute of the class is the ntElements attribute. This attribute is a Map of NtElement instances, where the element's identifier is used as a key. The NtElement class is characterised by its identifier and another String type attribute detailing the type of element in the network. Another attribute, instance of a HashMap class and termed connections, allows detailing information of the different connections the network element is involved in by referencing instances of the Connection class, using the connection identifier as a key in the map.

Connection objects are characterised by its connection identifier and the network where the connection is occurring, referenced by the attribute network, instance of the Network class. The endpoints involved in the connection are referenced in an attribute, an instance of the HashMap class, where the identifier of the network element acting as connection endpoint is used as a key. An attribute termed qoSparam, an instance of the QoSparams class allows specifying the required quality of service characteristics of the connection. Finally an attribute termed billingInfo, instance of the class connBillingInfo, allows keeping track of the connection duration and billing parameters according to its quality of service and endpoints. This information will be used to update the billingInfo attribute of the corresponding User instances involved in the connection when this is finished.

For this initial generic model of a network attached to FlexiNET architecture two basic network elements have been defined as specialization of the NtElement class, terminals and access nodes. The FAccessNode class tries to model the FlexiNET access nodes, connecting a network to the FlexiNET architecture. Since an access node exists per Network instance, a dedicated attribute in the Network class, fAccessNode, has been added. The access nodes will require persistency for data parameters dependent of the network the access node belongs to (UMTS / WLAN) to be determined in the respective specialization elements (FUAN, FWAN respectively).

Network terminals are also defined as specialization of the NtElement class. They inherit the identifier, type, network and connections attribute of that class and present a user attribute referring to the terminal user as well as a the characterization of the terminal in terms of manufacturer and model with a couple of devoted attributes. An attribute termed software, allows referring to the software version the terminal is based on.

Multiple methods in each of the presented classes allow access and manipulation of the respective values in their attributes.

### 5.2.4. UMTS specialization from the generic model

The generic model provides a general base to represent the intended basic operational procedures and relations within telecommunication networks, having user and service information as main concern. Network specific needs make necessary the extension of the generic model in order to cope with the related network dependent parameters.

Figure 5.8 shows a decomposition of UMTS user profile information regarding its functionality. The amount of parameters involved in a single

UMTS user profile is overwhelming and, though an ultimate aim, it is not necessary to keep all them for demonstrating the basic model's purpose within FlexiNET, as enabler of persistency and common treatment for data of technological networks.

As shown in Figure A4.2 (Appendix 4), classes in the generic model have been extended to cope with the operational needs of UMTS networks. The UMTS_NtDepData represents data needs of a user in a UMTS network for basic operation: the ptmsi attribute keeps the packet-temporary mobile subscriber identity of the user as assigned by the corresponding UMTS network. The rai attribute keeps the routing area identifier for the user. A Boolean attribute lets track the attachment status of the user to the UMTS network and with two Date instance attributes allows to determine the time of ocurrency. Another attribute, a String array of 3 elements, allows keeping the authentication vector values for the user until they are refreshed.

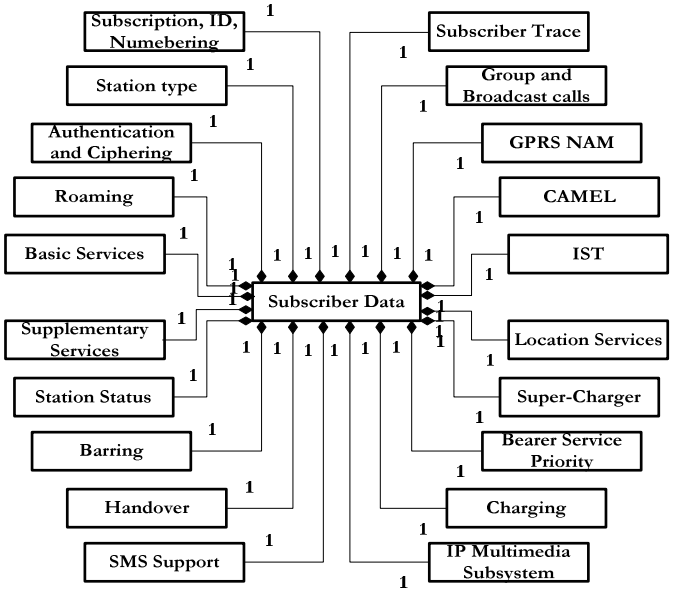



Figure 5.8 UMTS Subscriber related data

The UMTS_Subscriber Class is just an instance of the User class with an special constructor that assigns the international mobile subscriber identity (IMSI) identifier in UMTS networks to a user upon its creation as UMTS_Subscriber and populates its ntDependentData attribute with a UMTS_NtDepData instance associated to the network identifier of the network creating the subscription. This is done in such a way to cope

with the requirements of FlexiNET in which a user, subscriber of a specific network type, can use other technologically different networks, without requiring a modification or replication of its user data apart from that exclusively related to the operation in each network.

The class UMTS_Network extends the generic Network class with an HashMap instance attribute relating the different source local reference (SLR) and destination local reference (DLR) of an SCCP connection with the packet temporary mobile subscriber identity (PTMSI) assigned by the network to the a user in that connection. Apart from that attribute, another attribute termed users_UMTS relates the different users in the UMTS network with the assigned PTMSI.

The generic Connection class has been extended by the UMTS_Connection class modelling a PDP context and its attributes defined as String values: the concatenated pair formed by the SLR and DLR values defining the associated SCCP connection, the network service access point identifier (NSAPI), logical link control service access point identifier (LLC SAPI), and the adress for the packet data protocol (PDP). The constructor of this class assigns the transaction identifier for the UMTS contexts to the inherited connID attribute of the UMTS_Connection instance. Finally the class F_UMTS_AN model the data needs for the FlexiNET UMTS access nodes allowing the physical interconnection of UMTS networks to FlexiNET, allowing accessing the data repository and as defined by the demonstration scenarios.

It has to be stated that, at the time of writing, though several parts of the model have been developed and its persistency and operation tested independently in a local setup, an interface providing the functionality towards the elements of the model and hiding the operational details towards realisation of the different planned demonstration scenarios (access to a web server through UMTS/WLAN terminals with support on FlexiNET as data repository in different roaming scenarios) is still to be finished.

## 5.3. Alternatives to FlexiNET.

The initial architecture in the FlexiNET project has evolved considerably, blurring, at the time of writing (February 2005) some of the advantages of the initial proposal.

The initial architecture presented two distinct and generic interfaces to be used by any network attached to FlexiNET, each one with independent purposes, accessing the data repository through the data gateway node or accessing the services hosted by application services, as shown in Figure

5.7 The current state of the architecture proposal presents an extended DGWN, hosting functionality of the FLAS. Furthermore the interfaces towards the DGWN from the different access nodes are now network dependent, existing a definition for the access to DGWN from FlexiNET UMTS Access Nodes with terminology and parameters congruent with UMTS network needs. A similar interface will need to be defined for the WLAN case, where FlexiNET WLAN Access Nodes will need to access the data in the data repositories through the DGWN. In a similar way, other technologically different networks will need its specific interface to access the data through the DGWN.

From the point of view of the author of this thesis, this design is undermining the goals and applicability of the FlexiNET concept as "universal" network since the design of the DGWN is now dependent on the networks to be attached to it and so will need a new interface for any new technology to be added to FlexiNET. Loosing the network technology-transparency in the core element of the architecture turns FlexiNET in a quite more complex substitute for storage of specific data of the UMTS user profile.

The benefits of the architecture at such would only be those provided by an improved performance in the overall operations where those parameters are involved. This has still to be proved, but at the light of the performance of the mapping operations between the network signalling protocols and the specific messages designed for invocation of Web Services-based applications, presented in the previous chapter, this improvement is unlikely to happen.

At this time, FlexiNET seems to have become a mere UMTS user profile data repository. As such, FlexiNET has a clear competitor, the 3GPP's Generic User Profile (GUP) [5.14]. The 3GPP defines in [5.15], an architecture for homogenization and centralization of User Profile related data. As shown in Figure 5.9 two basic entities are the main functional components of the architecture:

- The GUP Server acts as a single point of access to the GUP architecture and performs functionality related to this access such as authentication and authorization.
- The Repository Access Function (RAF) hides the details of the access to the data repositories and provides a harmonized interface towards them.

Both Rg and Rp interfaces to GUP Server and RAF entities are based in web-services mechanisms (SOAP) [5.16].

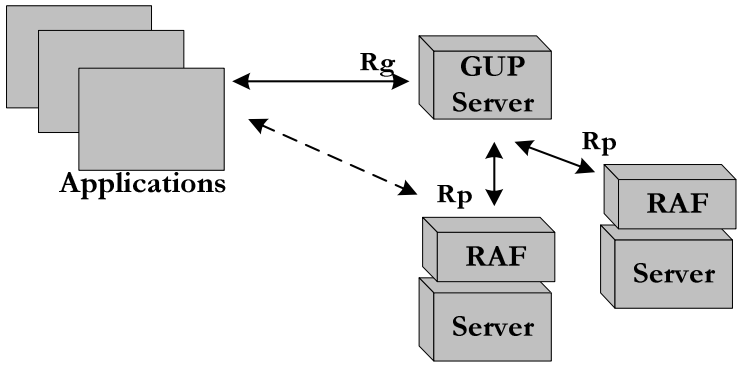



Figure 5.9 3GPP's GUP Architecture

The GUP data contents are defined as an open composite of different Profile Components, each of them with a unique identity and description accompanying a defined payload as shown in Figure 5.10. The profile, its components and attributes are syntactically defined in terms of XML schemas [5.17].

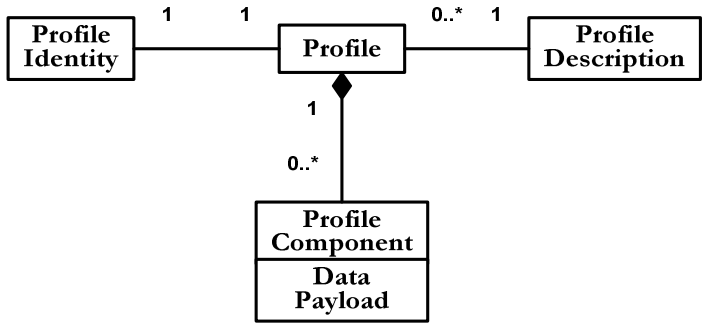



Figure 5.10 Data Profile as a composite of components

Comparing FlexiNET with the 3GPP's GUP architecture and profiles, it is possible to establish a clear analogy between the GUP server functionality and that of the FlexiNET Data Gateway Node (DGWN) and the application servers (FLAs) of the initial FlexiNET architecture. While the initial FlexiNET architecture tries to provide an "universal" environment for network storage and management of telecommunication networks, a lack of defined data access authorization policies might turn the architectural model impractical. Despite the access to FlexiNET data could be controlled by authorization mechanisms in the FLAS entities as a first step and initial requisite for subsequent data requests towards the DGWN,

this mechanism has not been conveniently treated so far in FlexiNET architecture. As a result, the use of FlexiNET by different operators might be impractical, since data regarding user's privacy, service subscription, etc would be accessible to any party connected to FlexiNET.

Another likely problem of FlexiNET is scalability. The DGWN, as a single physical entity acting as data provider to multiple networks within FlexiNET architecture with multitude of transactions per second, relative to multiple users, is bound to be exhausted. The distribution of functionality belonging to the DGWN in several physical entities or allowing different DGWNs per FlexiNET architecture could provide a solution to these escalability limitations. InterDGWN communication would be another issue to study, allowing the interconnection of different FlexiNET instances and with the proper security and authorization mechanisms, sharing and accessing information in a controlled-secure way. This would be congruent with other initiatives towards "federation" of networks and data access permission for the members of such federations, creating Personal Area Networks [5.18][5.19]or Ambient Networks [5.20] with the user's information as the base for interworking where this interworking is dynamically established without the need for pre-configurations or managerial negotiations between the network operators.

# 5.4. Summary

This chapter has presented architectural models to allow service and network interoperation by means of convergence in the signalling plane, thanks to common elements and procedures for handling service, user and operation data. An introduction to the evolution of data location and management in enterprise environments has been used as an initial analogy to introduce the subject in telecommunication networks. An ongoing project and its architectural initial proposal have been introduced as well as a data model to support the intended operations provided by that architecture and authored by the author of this thesis. Analysis of this architecture and its evolution has been provided as concluding steps for this chapter.

# 5.5. Chapter References


[5.1] Barker & Massiglia, "Storage Area network Essentials", John Wiley & Sons Inc.. ISBN: 0471034452.

[5.2] Shepler et al., "Network File System v4 protocol", IETF RFC 3530, April 2003.

[5.3] Microsoft Corporation, “Common Internet File System (CIFS) file access protocol”. Version 0.6a. March 2002.

[5.4] ANSI INCITS 351-2001, “SCSI primary Commands-2 (SCP-2) ”. 2001.

[5.5] ANSI INCITS 350-2003, “Fibre Channel Protocol for SCSI, 2nd Edition.”. 2003.

[5.6] Telikepali, Drwiega & Yan. “Storage Area Network Extension Solutions and their performance Assesment”. IEEE Communications magazine. April 2004.

[5.7] J. Soler & A. Fosgerau “Applying IT strategies as Interoperability Solution in Telecommunication Networks”. 7th International Symposium on Communications Interworking. November 2004.

[5.8] Rupp et al., “Flexible Universal Networks. A new approach to telecommunication services”. 8th World Multiconference on Systematics, Cybernetics and Informatics, July 2004.

[5.9] Russell et al., “Object Data management Group-3 Standard: The Object Data Standard”. Academic press, 2000. ISBN: 1558606474

[5.10] JSR 12, “Java Data Objects Specification, version 1.0.1” Sun Mycrosystems. May 2003.

[5.11] “Unified Modelling Language Specification, version 1.5”, Object Management Group Inc., March 2003.

[5.12] http://www.omondo.com

[5.13] http://www.eclipse.org

[5.14] 3GPP TS 22.240 v6.5.0 “Service requirements for the 3GPP Generic User Profile (GUP)”. January 2005.

[5.15] 3GPP TS 23.240 v6.6.0 “3GPP Generic User Profile (GUP)-Architecture”. December 2004.

[5.16] 3GPP TS 29.240 v1.0.0 “3GPP User Profile-Network”. December 2004.

[5.17] 3GPP TR 23.941 v6.0.0 “3GPP Generic User Profile (GUP)-Data Description Method”. December 2004.

[5.18] Olsen et al., “User Centric Service Discovery in Personal networks”, Proceedings of the International Symposium on Wireless Personal Multimedia Communications (WPMC) 2004. ISBN: 87-91696-21-6

[5.19] Sørensen et al., “A user centred methodology for establishing PN/PAN scenarios”, MAGNET Workshop on “Personal Adaptative Global Networks. Visions and Beyond”. November 2004.

[5.20] Niebert et al., “Ambient Networks: an architecture for communication networks beyond 3G”, IEEE Wireless Communications, April 2004.

# 6. Summary and Conclusion

The work herein presented has dealt with evolution of telephony services towards PSTN/IN and Internet convergence.

The increase in use and penetration of Internet during the last decade, driven by the use of the World Wide Web, inevitably is affecting the traditional telecommunication business and service models. Users are demanding exploitation of the new capabilities, the new IP-based environments allow: multimedia contents, ubiquity, network and application interoperability. Packet-based transmission techniques provide a common platform for unifying traditional voice and data services in a common infrastructure, minimizing maintenance and management effort multiplication and enabling reducing costs. Furthermore, the generalization of deployment of 3rd generation mobile networks, enabling IP-based services and the need of operators to recover the huge investments made on them, will hopefully trigger in the years to come the deployment of innovative services and foster the evolution of new technologies and mechanisms enabling them.

The initial chapters of this thesis have presented the different mechanisms for provision of special features complementing the basic voice service both in traditional switched telephone networks and VoIP networks, outlining the basic architectures of the Intelligent Network in the case of PSTN and H.323 and SIP in the case of voice over IP (VoIP) deployments.

Hybrid architectures, as a synergic evolution of the traditional and new telephony environments have been introduced and the work developed within the ISP GEMINI project presented. Within that project's architectural proposal, the author designed, developed and tested telephony services based on a SIP environment with the support of SIP Servlets technology, accessible within that environment by IP-based terminals and, thanks to the proper gatewaying mechanisms provided by the architecture, accessible also by PSTN terminals in a conventional PSTN/IN environment.

The need for decoupling service logic invocation and execution from network dependent mechanisms was presented afterwards. The conception of the GIN Servlets, as an application of SOAP based mechanisms to the GEMINI architecture with that purpose, has been detailed. The set of services that this mechanism enables have been demonstrated, with the development and testing of GIN-based services. The interest of the GIN mechanism lays in its independence from any existing service model or service mechanisms, comparing it with current industry accepted proposals such as X-Parlay providing a Web Services interface to Parlay based service architectures. The limited scope of target services for the defined GIN requests and responses, make the proposal not consistent as a general purpose mechanism for decoupling service logic from network operations. Nevertheless, the method can be generalized and extended to other service sets, by studying its specific messaging needs and designing the proper message elements and exchanges as it has been done in the case of GIN.

Finally, a brief introduction to the IST project FlexiNET and the concept of service and user data centralization, to enable interoperation among networks and minimize signaling traffic within a single network, has been provided. Along with it, a generic data model defining the relation between different elements and allowing basic network operation processes in the data-centralizing FlexiNET overlay architecture, has been introduced as well as the author's point of view about the validity of the current FlexiNET architecture, i.e. scalability issues, data use control, access to other FlexiNET instances (federation).

A number of issues related to telephony services and service deployments have not been dealt with in this work, i.e. service features interaction, service coordination, service access authorization, etc. The work performed and the decisions taken had been done in accordance to the punctual requirements of the projects where these works were held, i.e. IST GEMINI and IST FlexiNET. While this last project is still an ongoing work and the efforts towards its aims are not concluded and so the conclusions presented not definitive, the results and work performed in the GEMINI project were innovative at its time and succeeded in demonstrating the capabilities for evolution in the area of telephony services.

New technologies broke in the telephony arena in the last stages of this works, i.e. peer to peer technologies (P2P) and its applicability to VoIP with the commercial success of Skype[5]. Applicability of these technologies to the realm of services and service architectures, synergies between

[5] http://www.skype.com/

them and already existing ones, i.e. P2P, SIP and Web Services, may be of key importance for the future evolution of the subject.

# Appendix 1. H.323 basic operation

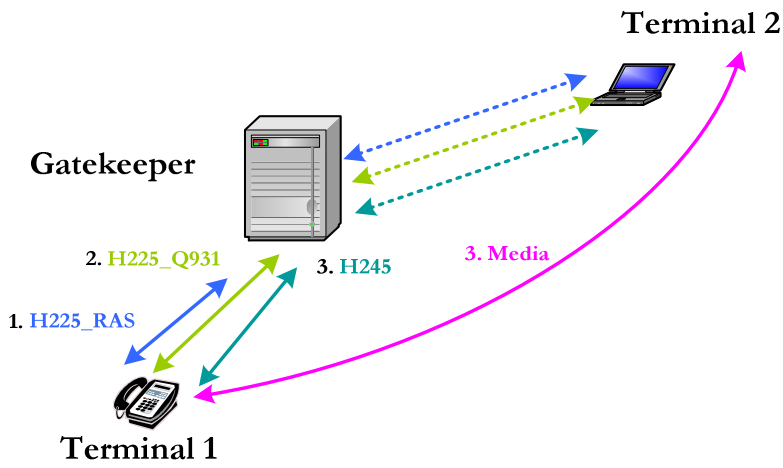



Figure A1.1 Basic H.323 architecture and operation

While H.323 terminals can communicate directly among themselves, H.323 configurations are based on a Gatekeeper acting as entities and communications manager.

As shown in Figure A1.1, initially terminals contact the Gatekeeper entity in order to access the H.323 "zone" the Gatekeeper is responsible for. The Gatekeeper acts for the terminals as registration, admission and status (RAS) server. This communication is done by a dedicated set of message flows and procedures within the H.225 protocol for that purpose (H.225.RAS).

Once a terminal has been granted access to the H.323 zone and has registered itself in the Gatekeeper, it can initiate a call to other H.323 entities. In order to do that, the call set-up procedures of H.225, inherited from those of Q.931 (H.225_Q.931 in the figure) are used to setup the call with the remote target terminal. While this signaling flow could be routed towards the receiving terminal directly, passing it through the gatekeeper enables to control the call status and to act upon this information (i.e. billing, triggering of services).

When the call is set-up, a negotiation of the characteristics of the media exchange starts, by means of H.245 flows and procedures, allowing identifying the capabilities of the involved terminals, agree on the media to be exchanged (media type, encoding, packetization) and start its transmission. H.245 flows use to be contained within H.225_Q.931 flows to minimize the ports required per H.323 connection (H.245 tunneling).

# Appendix 2. SIP basic operation

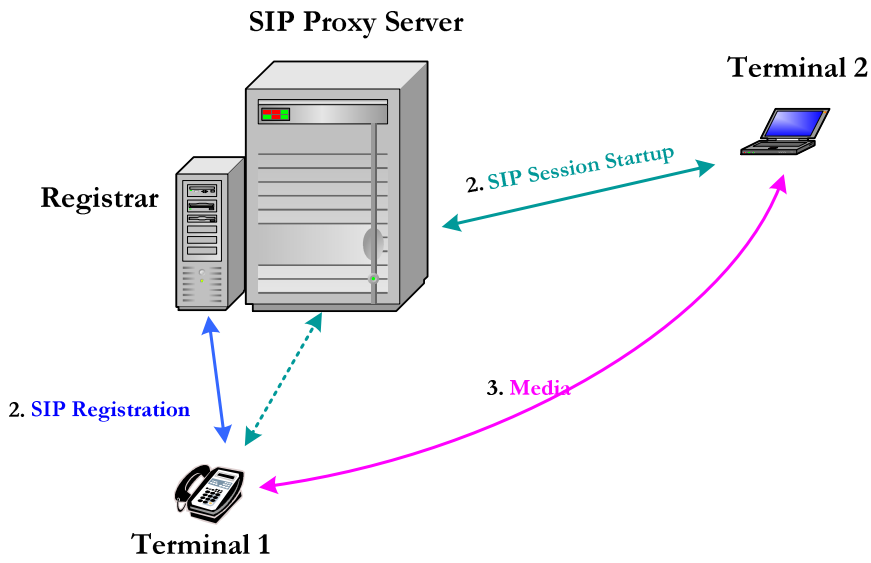



Figure A2.1 Basic SIP architecture and operation

As in the H.323 case, even though SIP terminals can communicate directly among themselves, usual configurations are based on the support of a SIP Server (Proxy or Redirection) that forwards or redirects incoming request to a user towards its actual location.

This mechanism is based on the help of a registrar entity, usually collocated with the proxy/redirection server. Initially, users register in the Registrar entity its current location. The access to the registrar may be preceded by authentication.

When a terminal wants to start a call, it sends an INVITE SIP method to the server where the receiver user is registered. This message contains a description of the capabilities of the initiating terminal, as well as a description of the contents intended for the call. The SIP INVITE method arrives to the proxy/redirection server. In the case of a proxy server, the INVITE message is forwarded to the current location of the receiver user. In the case of a redirection server, this location is returned to the initiator, so that it can contact directly with the receiver in its current location.

The example shown in the figure, displays the case in which a proxy SIP server is forced to remain in the signaling between the communicating terminals: once the SIP INVITE message arrives to the receiver, a three

way handshake is completed with the initiator, in order to agree with the characteristics of the call.

After this handshake is completed, the media can flow among the terminals until completion of the call. The SIP signalling associated to the call termination, will also pass through the proxy, enabling acting accordingly (i.e. billing, service triggering).

# Appendix 3. SIP extensions (December 2004)

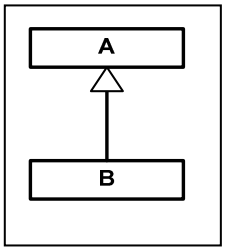
| RFC | Enhancement | Purpose |
|---|---|---|
| **RFC 2976** | INFO method | Mechanism to carry application layer information, related to the SIP session. |
| **RFC 3265** | SUBSCRIBE and NOTIFY methods | Mechanism to subscribe and notify specific events. |
| **RFC 3262** | PRACK method | Mechanism to introduce reliability in provisional responses. |
| **RFC 3311** | UPDATE method | Mechanism to update parameters of a session without modifying state of dialog. |
| **RFC 3325** | P-Asserted-Identity and P-Preferred-Identity header fields | Mechanism allowing SIP proxies to add user identity information and callers to request privacy. |
| **RFC 3428** | MESSAGE method. | Mechanism allowing the transfer of instant messages. |
| **RFC 3326** | Reason header field | Mechanism to include additional information regarding a request. |
| **RFC 3327** | Path header field | Mechanism to register a list of proxies between user agent and registrar. |
| **RFC 3313** | P-Media Authorization header field | Mechanism to integrate QoS admission control with call signaling. |
| **RFC 3486** | Comp parameter in URI and Via header. | Mechanism to indicate compression needed and compression details. |
| **RFC 3515** | REFER method.<br>Refer-To header.<br>Refer event package. | Mechanism allowing referring requests and receiving notification of a referred request. |
| **RFC 3581** | Rport parameter in Via header. | Mechanism allowing symmetric response routing in configurations involving |

| | | |
|---|---|---|
| | | NATs. |
| **RFC 3608** | Service-Route header field | Mechanism to indicate a route towards a service point in response to a registration. |
| **RFC 3603** | P-DCS-Trace-Party-ID, P-DCS-OSPS, P-DCS-BILLING-INFO, P-DCS-LAES and P-DCS-REDIRECT headers. | Mechanism allowing the exchange of customer and billing information within a specific architecture (Packet Cable) |
| **RFC 3680** | Event package for registrations. | Mechanism allowing the subscription by watchers to modifications in the registration state of endpoints. |
| **RFC 3824** | Event template package for watcher information (winfo) | Mechanism allowing presence authorization. |
| **RFC 3856** | Event package for presence (presence). | Mechanism allowing SIP to behave as a presence protocol. |
| **RFC 3842** | Event package for message waiting and message waiting indication (presence). | Mechanism allowing SIP to behave as a presence protocol. |
| **RFC 3841** | Accept-Contact, Reject-Contact, and Request-Disposition header fields. | Mechanism allowing callers to specify preferences about the request handling in servers. |
| **RFC 3840** | Extension of Contact header field. Extra tags (+). | Mechanism allowing user agents to indicate its capabilities to other user agents. |

| | | |
|---|---|---|
| **RFC 3891** | Replaces header. | Allows replacing one dialog with another in multiparty sessions. |
| **RFC 3892** | Referred-by header, referred-by token 429 response, | Mechanism allowing to provide information about a refer request to the target of the request. |
| **RFC 3893** | Authenticated Identity Body MIME body format. | Mechanism to derive integrity and authentication properties form digitally signed messages or message fragments. |
| **RFC 3911** | Join header. | Mechanism allowing the creation of multiparty SIP sessions. |
| **RFC 3903** | PUBLISH method, 412 Response,<br>SIP-ETag, SIP-If-Match header fields. | Mechanism allowing publication of event state. |

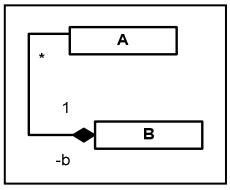
# Appendix 4. FlexiNET's (initial) Data Models

# A4.1 Guide to the UML-diagrams semantics



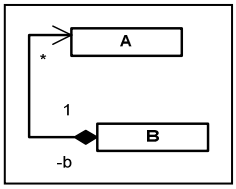

**Inheritance**

Class B is an specialization of class A, and inherits its characteristics (attributes and methods).




**Composition Association (i)**

The association relation is navigable in both directions: Class A has a single component instance of Class B, called b. Class B, may have multiple components, instances of class A.




**Composition Association (ii)**

The association relation is navigable only in the direction from B to A: Class A has no component instances of Class B (the name and multiplicity are meaningless). B has may have multiple components, instances of Class A.

When the rhomb shape at the edge of a composition association is filled, the life-time of the class instance associated at the other side of the association is bounded by the lifetime of the instance at the rhomboid side. On the other hand, when the rhomb is empty, the life times of the associated class instances are independent from each other.

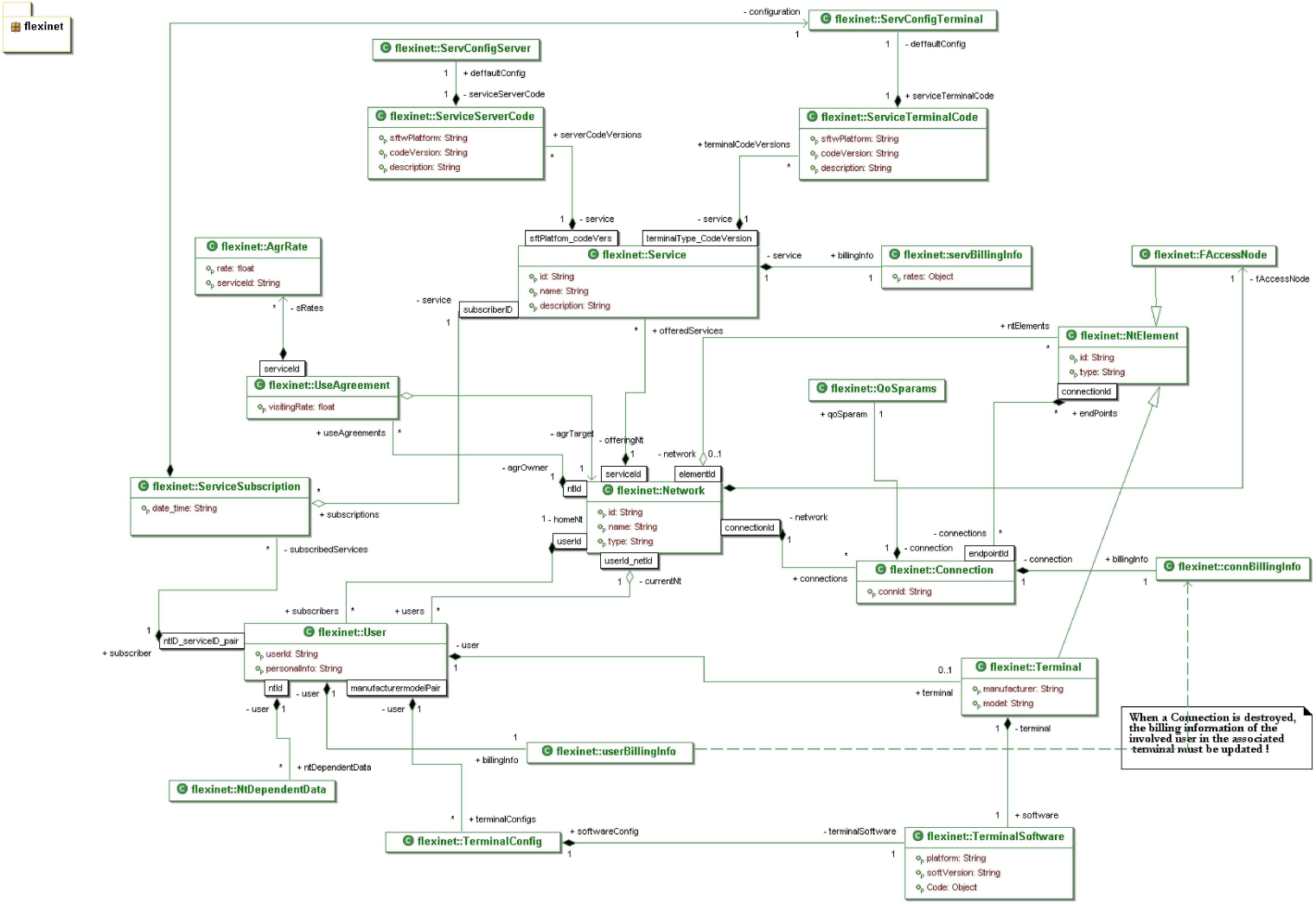



Figure A4.1. Generic FlexiNET data model

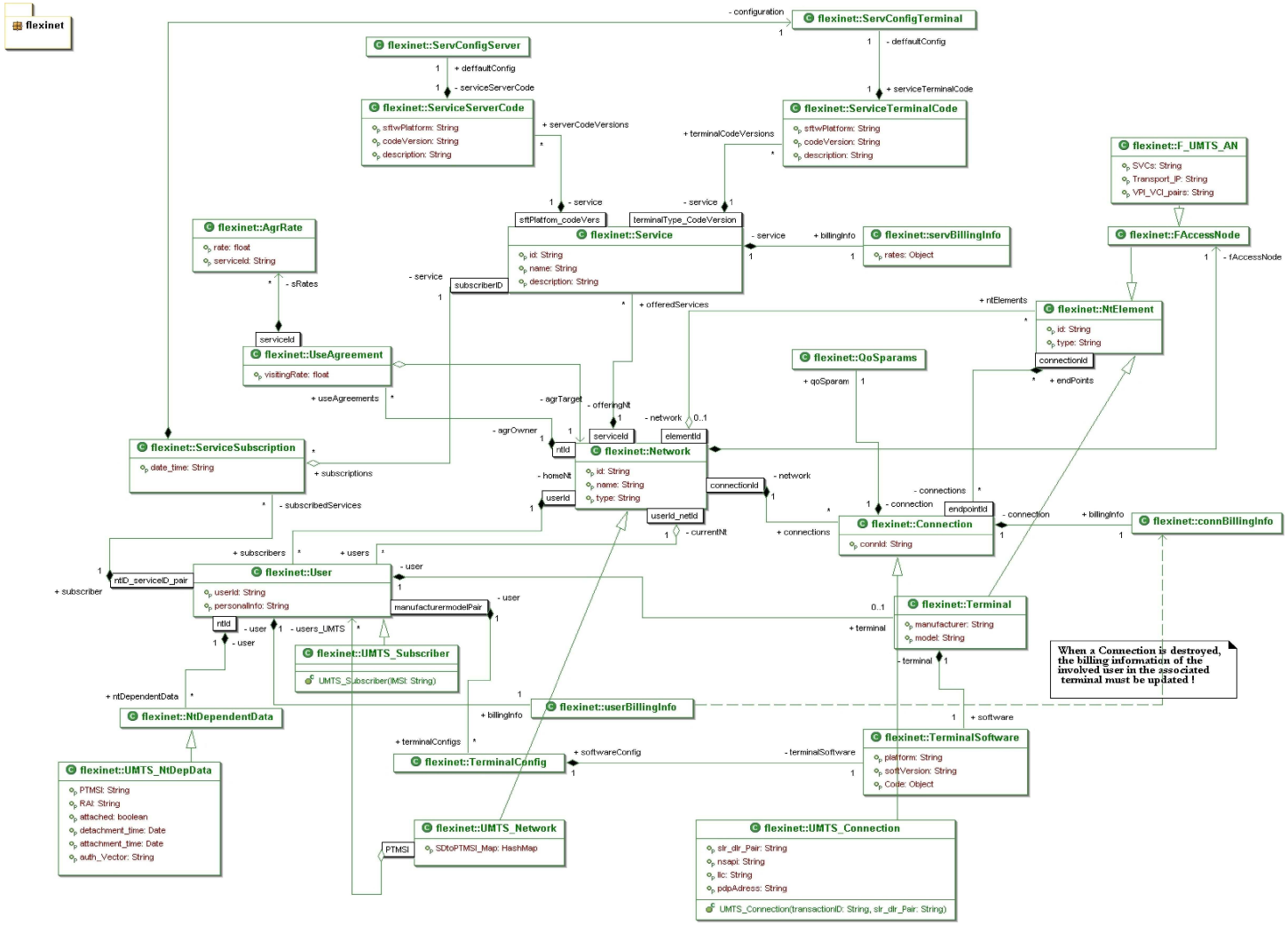



Figure A4.2 Generic FlexiNET data model and UMTS extensions

*Explicit Opus Est*

JS. (May 2006)