KUNGL
TEKNISKA
HÖGSKOLAN

Preserving Integrity in Telecommunication Networks Opened by the Parlay Service Interface

Magnus Almkvist and Marcus Wahren

The Royal Institute of Technology, Stockholm, Sweden
Department of Microelectronics and Information Technology
2002.09.30

Preserving Integrity in Telecommunication Networks Opened by the Parlay Service Interface

Magnus Almkvist and Marcus Wahren
magnus@almkvist.net marcus@wahren.org

A thesis
presented at the Royal Institute of Technology
in fulfilment of the thesis requirements for
the degree of Master of Science in Electrical Engineering

Examiner
Prof. G. Q. Maguire Jr.
KTH/IMIT
Isafjordsgatan 39
SE-164 40 Kista
maguire@it.kth.se

Supervisor
Karl-Gunnar Eklund
Skanova
Vitsandsgatan 9
SE-123 86 Farsta
karl-gunnar.b.eklund@skanova.com

This thesis is publicly available at:
ftp://ftp.it.kth.se/Reports/DEGREE-PROJECT-REPORTS/
020930-Magnus-Almkvist-and-Marcus-Wahren.pdf

Abstract

This Master's Thesis in Electrical Engineering concerns the introduction of a Parlay gateway in Skanova's public circuit switched telephone network, what network integrity problems this brings, and how to preserve the integrity of the network.

There is a rising demand from the market on Skanova to be able to offer integrated and useful services via their network. Examples of such services are *Web Controlled Call Forwarding* and *Virtual Call Centres*. Until now, these services have been implemented with the *Intelligent Network* concept which is a technology for concentrating the service logic in the telephone network to centralised service platforms within the network operator's domain. Developing new services in this environment is expensive and therefore, Skanova wants to open the network for third party service providers. The opening of the network is enabled by the introduction of a gateway implementing the open service interface *Parlay*.

The crucial point when opening the network for third party service providers is to maintain the integrity of the network. Parlay is an object oriented *Application Programming Interface* that enables a third party service access to core network resources in a controlled manner.

The authors' definition of network integrity is: "the ability of a network to steadily remain in a safe state, while performing according to the expectations and specifications of its owner, i.e. delivering the expected functionality and providing means to charge for utilised network resources".

The thesis describes a few services implemented via the Parlay interface and points out examples of activities in these services that may jeopardise the integrity of the network. The described activities belong to one of the two categories: *Call Control Functionality* or *Lack of Charging Instruments*.

The thesis also describes two important methods for addressing encountered integrity problems. The methods are: *Parlay Service Level Agreement* and *Policy Management*.

Finally, the solutions are compared and the conclusion is that *Policy Management* is a conformable and flexible method for addressing lots of integrity problems and that these are important qualities, since new integrity problems will arise all the time.

Keywords: Parlay API, network integrity, policy management, Parlay SLA, PSTN, charging, telephony, telecommunication network, Parlay gateway, intelligent network

Acknowledgements

We would like to thank our employer Kerstin Erlandsson and our supervisor Karl-Gunnar Eklund for giving us the opportunity to do this thesis at Skanova and for their kind help in all situations. We have gained a lot of knowledge about telecommunication and service development. At Skanova we also would like to thank Oscar Bravo for his valuable technical contribution, and Åke Hedevärn for his good hints when writing this report.

Many thanks to Thomas Svensson at Incomit for his heavy commitment in making us acquainted with the concept of Policy Management. We would also like to thank Petter von Dolwitz at Appium for his willingness to answer our questions.

Finally, we would like to show our sincere gratitude to our examiner Professor G. Q. Maguire Jr., at the Royal Institute of Technology, for his valuable and momentous comments on this report, as well as his exceptionally fast e-mail replies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This report is the result of a thesis project forming the last part of the Master of Science degree in Electrical Engineering at the Royal Institute of Technology in Stockholm, Sweden. Our employer for the project is Skanova[1].

1.1 Background

Skanova is the dominant telecommunication network provider in Sweden. They offer telephone services with different degrees of refinement to telephone operators. The market continuously demands that new services are added to the network. Implementing those new services in Skanova's network is often time consuming and complicated because new services need unique adaptations of the network infrastructure. Therefore, Skanova has decided to add a new component to their network, a Parlay gateway[2] that is using the Parlay interface (see section 4). The Parlay interface is a standardised and open application programming interface. By separating the service logic from the switching logic, this gateway will enable third party Application Service Providers (ASP) to develop and add their own services to Skanova's network (see section 3).

1.2 Problem Definition

The objective of this master's thesis is to focus on the integrity problems that arise when a public switched telephone network is opened up for third party service access via a gateway implementing the Parlay interface. This is an important issue that a telephone network operator should inquire into before implementing a Parlay gateway in his network. To fulfil the objective the following issues must be addressed:

- Determine different types of undesirable network states, requests or actions.
- Determine methods for preventing undesirable network states, requests or actions.
- Determine how to handle charging in different situations.

[1] Skanova is a secondary name for Telia AB.
[2] Internally this gateway is named *Systemaccess Parlay*.

1.3 Method

To be able to solve the problem stated above the following programme was adopted:

1. Define *network integrity*
2. Find different categories of integrity problems
3. Propose possible solutions to the problems in each category
4. Compare the solutions

These points will be accompanied by a number of descriptions of different techniques (e.g. Parlay API, SS7, IN, policy management, etc.) that are necessary to understand.

1.4 Limitations

Finding all integrity problems is a complex task since there is a considerable number of areas where the network integrity could be harmed. We do not intend to be complete with respect to integrity problems, or their solutions. Therefore we state the following limitations:

- We will only analyse functional integrity problems related to the areas *Call Control* and *Charging*.
- We will only give examples of categories of integrity problems. These examples will be based upon example services.
- Hardware related integrity problems are out of scope of this report.
- We will only study Parlay API version 3.0. There are newer versions[3] which possibly lead to other solutions.
- We will focus on the Parlay Generic Call Control API because this is the only API currently supported by the Skanova network and it is used by most service applications.
- We will only propose solutions based upon existing technologies.
- When recommending solutions, we will not take economic consideration.

[3]Parlay API version 4.0 is under development.

Chapter 2

The Skanova Telephone Network

This chapter describes the systems that enable telephony and gives the reader the technical prerequisites for understanding the Skanova telephone network infrastructure. If needed, we would like to call the readers attention to the list of acronyms in appendix A.

2.1 Telephone Network

The telephone network consists of two main parts: the transmission and the signalling network. The two networks are logically separate, but interconnected in several nodes. The transmission network carries the voice data and the signalling network is used for control communication between different network entities such as exchanges and service entities. Throughout this report we will mean both the signalling and the transmission network (the composition), when we are talking about the telephone or telecommunication network.

2.1.1 Fundamental Telephone Network Structure

The telephone network is organised in a hierarchical structure. Please refer to figure 2.1 on page 4. In Skanova's network Sweden is divided into thirteen transit areas, which constitutes the highest level of the structure. The transit areas are interconnected via *transit exchanges* [Jarsjö, 1992].

The transit exchanges are Ericsson AXE stations [Rydqvist, 1987]. No subscriber line accesses are connected to these exchanges. Instead, the transit exchanges interconnect the transmission circuits coming from other transit exchanges and *local exchanges* (described below). The transit exchanges are able to use the *Intelligent Network Application Protocol* (INAP), see section 2.1.4 on page 7. Thus, the transit exchanges are *Service Switching Points* (SSP), because they are able to communicate with service platforms in the network, so-called *Service Control Points* (SCP). This is described more in detail in section 2.2.

To make the network reliable and redundant, all transit exchanges are organised in pairs and if one transit exchange fails, the other takes over. One transit exchange serves one transit area which is divided into one or more numbering

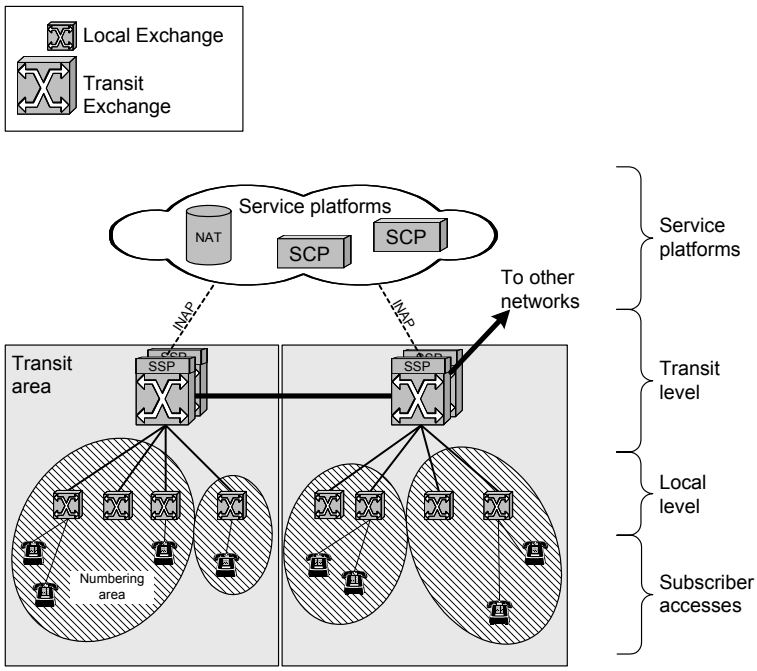


Figure 2.1: The hierarchy of the network

areas. However, there are exceptions from this in big cities where several transit exchanges may serve one numbering area.

Below the transit level is the local level, which consists of local exchanges. These are Ericsson AXE stations as well. All local exchanges in a transit area are connected to same transit exchange serving that area. A numbering area is served by one or more local exchanges. Closest to the subscribers, in the hierarchical structure are, the subscriber access lines that are connected to the local exchange [Hansson and Teglöf, 1985].

The local exchanges handle the logic for each subscriber access line. This includes information about which telephone operator a subscriber access line has, settings for supplementary services, etc. Charging information, *Toll Tickets* (TT) [Valas, 1999], for outgoing calls are also generated here. These toll tickets are utilised by post processing systems [Henström et al., 1992] and later used for billing.

To connect Skanovas network with other networks such as fixed and mobile networks and networks of other operators there exist special exchanges. Attention has to be paid to ensure that these exchanges handle charging and signalling with these other networks in a correct manner.

2.1.2 Transmission Network

The network carrying the voice data is called the transmission network. The transmission network is currently based on circuit switching technology, which implies that a dedicated path is set-up between communicating entities such as

exchanges. The path set-up is transparent, as if the endpoints where directly connected to each other.

The carrier medium is fibre and the transmission protocol is *Asynchronous Transfer Mode* (ATM) [Stallings, 2000], which conveys the data in fixed-size packets called cells. Logical circuits in ATM are called *Virtual Circuits* (VC). The VCs are organised as shown in figure 2.2. For each voice data connection, one VC with a transmission capacity of 64 kbps is reserved. This ensures sufficiently high quality for the call, but the path is rarely fully utilised.

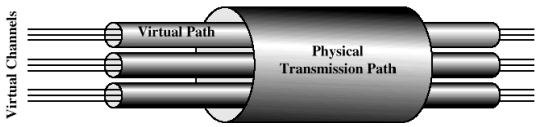


Figure 2.2: The logical connections in ATM [Stallings, 2000]

2.1.3 Signalling Network

The signalling system, named *Signalling System No 7* (SS7), in the telephone network is logically separated from the transmission network. This implies that the signalling traffic can be sent over any medium, e.g. the transmission network in so-called *Permanent Virtual Circuits* (PVC). The communication in the signalling system is packet switched and similar to the Internet, but with a completely different protocol stack (see figure 2.3) [Boman et al., 1992].

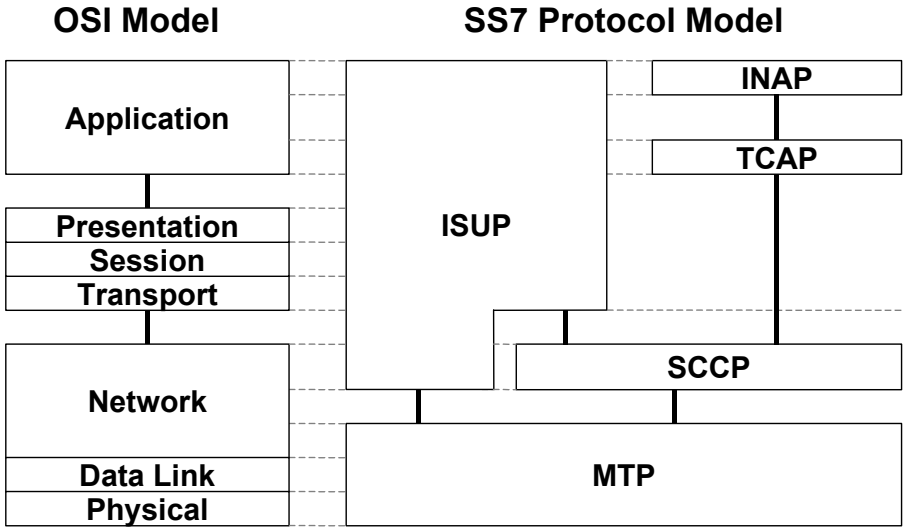


Figure 2.3: The SS7 protocol stack

The signalling system interconnects network entities such as exchanges, SSPs, SCPs, etc. The signalling network is used for call-establishment, service control functions, charging, routing, and information exchange functions in the network.

The signalling links are interconnected through *Signalling Transfer Points* (STP). The STPs are packet switches for the SS7 network. They receive and route incoming signalling messages to their destination. They also have routing functionality such as load balancing, ability to determine new routes, etc.

The STPs are organised similar to the transmission network in a hierarchical structure. On the transit level there are transit STPs and on the local level there are local STPs. Below this level are the end points of the signalling network; they are called *Signal End Point* (SP). Each SP is connected to two local STPs. All local STPs are connected to two transit STPs. The transit STPs are organised in two halves, right and left. All the STPs in one half are interconnected to each other. Corresponding STPs in each half are interconnected via double links. The links between the halves are only used as a last resort in case of fault or overload. Figure 2.4 illustrates the structure of the signalling network.

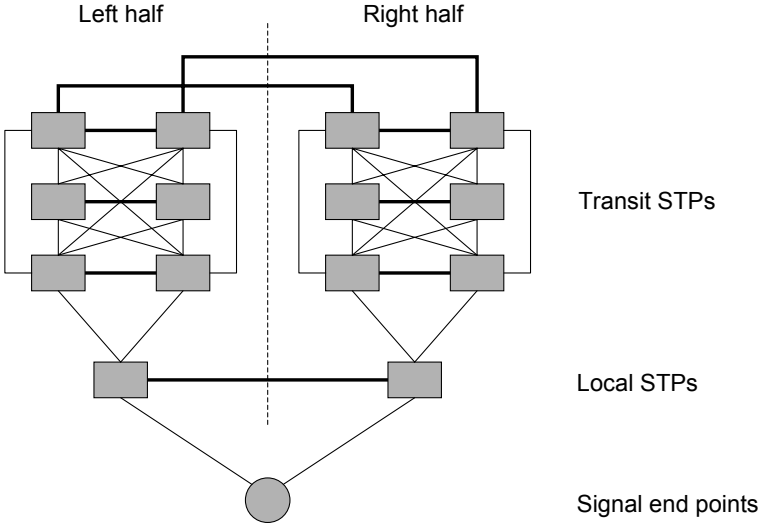


Figure 2.4: The hierarchy of SS7

2.1.4 Signalling in SS7

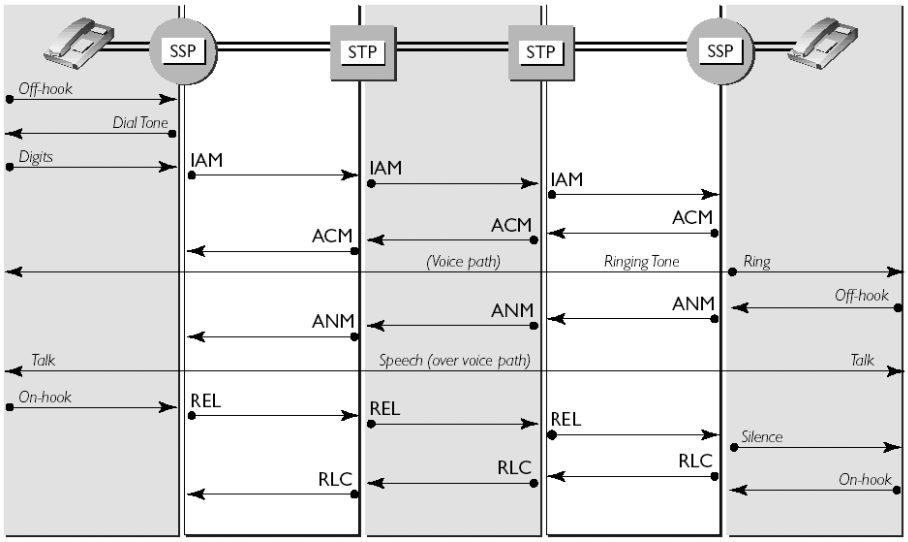
The protocol stack used in the Signalling System No 7 is illustrated in figure 2.3 on page 5. It consists of several protocol levels that correspond to different levels in the OSI model. It is designed to both facilitate functions for call set-up, network management, and maintenance.

The *Message Transfer Part* (MTP) [Modarressi and Skoog, 1990] is the link-layer protocol of SS7. It ensures that two signalling points can exchange messages reliably. It has functions for error correction and flow control. MTP also implements some network-layer functionality such as node addressing, routing, and congestion control (reconfiguration of the signalling network in case of congestion or blockage).

The *Signalling Connection Control Part* (SCCP) [Boman et al., 1992] provides two major functions that MTP does not: the ability to address applications within a signalling point and the ability to send other forms of messages than

call set-up, i.e. messages for IN-services (IN stands for Intelligent Network, see section 2.2). The SCCP protocol has mostly network-layer functionality.

The *ISDN User Part* (ISUP) [Modarressi and Skoog, 1990] provides the ability to set-up, manage, and release trunk circuits that carry voice and data. Despite its name ISUP is used both for ISDN and non-ISDN calls. The flow-chart in figure 2.5 illustrates how ISUP is used to set-up a call.



1. The Initial Address Message (IAM) contains the dialled number and call set-up information. It is sent to the called party.
2. The Address Complete Message (ACM) indicates that the called party is available and that the voice circuit is reserved.
3. The Answer Message (ANM) is sent when the called party answers. It initiates the generation of charging information.
4. The Release Message (REL) indicates that the circuit is released by any party.
5. The Release Complete Message (RLC) confirms the Release Message and terminates the charging process.

Figure 2.5: Simple call set-up via ISUP [ISUP, 2002]

The *Transaction Capabilities Application Part* (TCAP) corresponds to the lower part of the application-layer [Modarressi and Skoog, 1990]. TCAP enables applications to invoke procedures in another part of the network and exchange the results. TCAP directly uses the functionalities that SCCP offers.

The *Intelligent Network Application Protocol* (INAP)[ETSI, 1994] provides the means to perform high-level service related communication. It uses the SCCP connectionless service as a transport protocol. For example, applications on an SSP communicate via INAP with SCP applications to realise services. The mechanisms behind these services are described in section 2.2. The currently used version of INAP in Skanovas network is called *Capability Set 1* (CS1) [ETSI, 1994]. Within a few years, a more powerful capability set, CS2, will be implemented in the signalling network. It will enable dynamic set-up and tear-down of multi-

party calls, telephone conferences, etc. Today such services can not be realised with functions in the network. This results in work around that make call set-ups that are inflexible and resource demanding, since they sometimes have to make use of double transmission circuits, so-called *tromboning*.

2.2 Intelligent Network

The *Intelligent Network* (IN) [IN, 2002] is the current telecom approach to provide value-added services in the telecom network. The concept is to move the service logic of the network from the switches to separate centralised platforms, so-called SCPs (SCP), see figure 2.6.

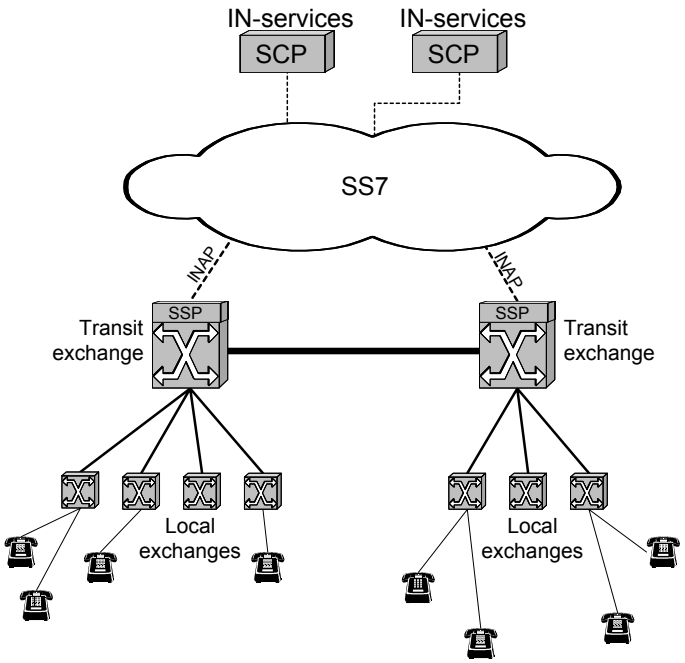


Figure 2.6: The Intelligent Network concept

The IN-applications use the SS7 INAP protocol (see section 2.1.4) for communication. This centralised structure, i.e. the transfer of service out of the exchange, makes it simpler to implement and modify services. Since the introduction of IN a range of new services have evolved. Examples of services are free phone (the called party pays for the call), virtual call centres, number portability (see section 2.3), voice mailboxes, etc. Figure 2.7 on page 9 shows an IN call with an interactive service request.

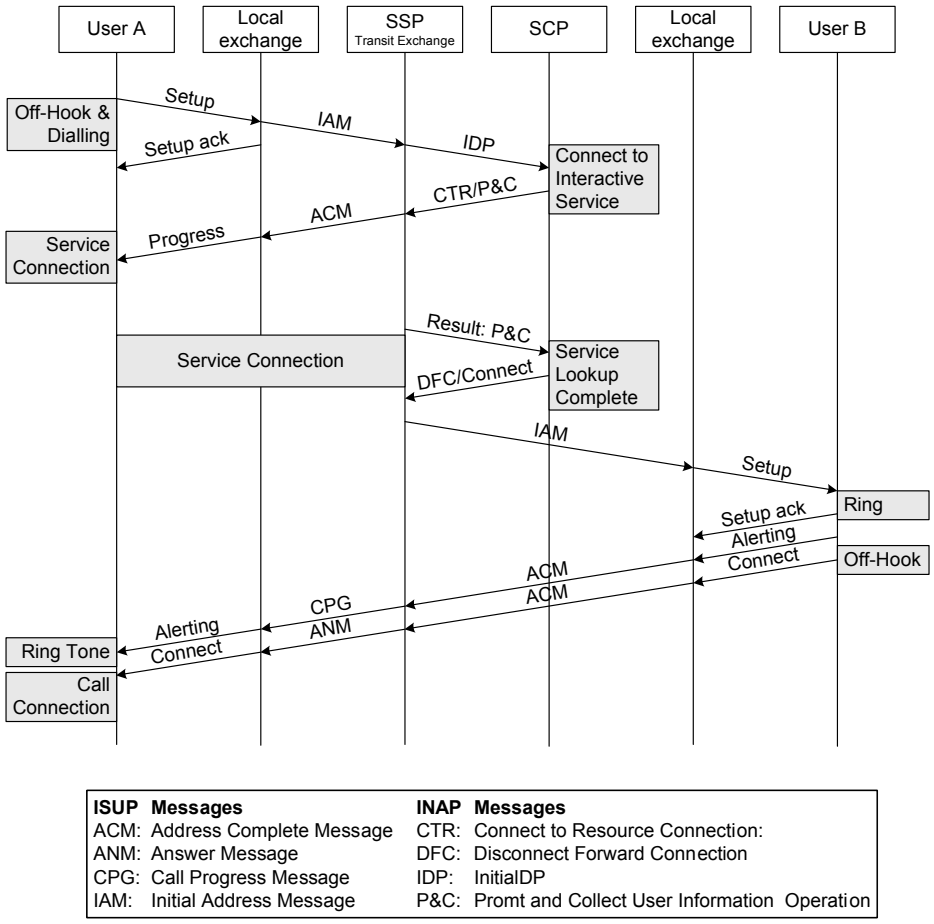


Figure 2.7: IN call with interactive service request

2.3 Network Address Translator

Number portability gives subscribers the possibility to keep their telephone numbers when moving within the numbering area or changing network operator [Albertsson, 2001]. Every subscriber number[1] formally belongs to a geographical area inside a certain numbering area. The network has a hierarchical structure where the calls are routed to the destination via exchanges that are responsible for different areas, and this is done by analysing the dialled subscriber number.

Some years ago number portability was not feasible because routing on the dialled number was the only method of determining where to terminate the call. Today number portability is enabled in Skanovas network via the *Network Address Translator* (NAT) and *All Call Query* (ACQ) [Orava, 2001]. The NAT is a SCP that contains a database with all ported subscriber numbers and a prefix

[1]By subscriber number we mean the ordinary number associated with a telephone subscription at home or at a company without a *Private Branch Exchange* (PBX), i.e. a non-service number.

pointing to the new exchange to which the ported number belongs. The NAT is shown in figure 2.1 on page 4.

All numbers dialled have to be checked to see if they are ported to a new exchange within the numbering area and this method is called ACQ. A query (from a local or transit exchange) is addressed to the NAT via the INAP protocol. Normally, only the transit exchanges and SCPs are able to communicate via INAP. The local exchanges are, however, equipped with *Light Weight INAP* (LW-INAP) [Albertsson, 2001] to be able to communicate with the NAT.

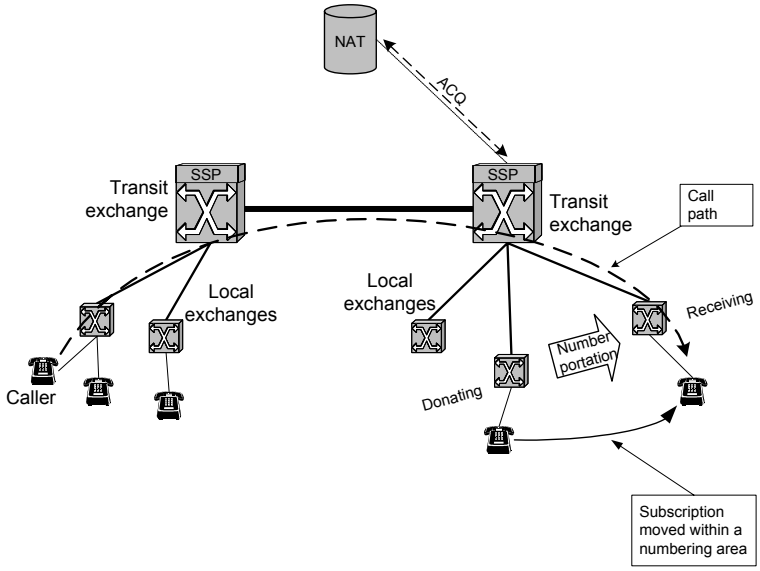


Figure 2.8: Number portability with ACQ

To be able to know when a dialled number is complete and when to send a query to the NAT the number length has to be known. The local exchange has information about the number length for all number series in its own transit area. For calls made between transit areas the number length is determined by an analysis in the terminating transit exchange as illustrated in figure 2.8. This implies that the local exchange has to do the query to the NAT for calls within the transit area, and the terminating transit exchange for calls originating outside the transit area.

Today it is only possible to port a subscriber number within a numbering area because the same number (without the area code) may co-exist in several numbering areas. Consequently, to be able to port a number outside a numbering area, e.g. common number length has to be introduced for the whole country. This is a complicated operation and will require a decree from The National Post and Telecom Agency (PTS).

Using NAT it is also possible to port a subscriber number to a service number (virtual call centres, free phone, premium rate service, etc.). In those cases, NAT redirects the incoming and ported number to the SCP where the service is implemented. At the SCP a decision is made where to direct the call.

2.4 Charging

One of the fundaments of operating a telephone network is to be able to charge users for the resources used. The operator has to keep track of starting time, duration of the call, resources utilised in the network, call origin, and destination. The subscribers are charged in different ways by means of this information. The following concepts are essential when handling charging.

Toll Tickets (TT) [Valas, 1999] are generated by and stored in the exchanges. They contain call starting time, duration, number of caller, dialled number, and also other information such as call forwarding (if any) and length indicator for calling and called number. The TT is created when the dialled subscriber answers and the voice path is established. When the call ends the TT is updated and completed. If a call lasts for a period of more than five hours the current TT is completed and a new TT is created for the same call, and so on. This guarantees that not more than five hours of charging information is lost in case of a failure and that *very* long calls are chargeable.

The stored TTs in the exchanges are regularly collected (every thirtieth minute) by post processing systems. The post processing system analyses the TTs and converts them into a standardised format called *Call Data Record* (CDR) [Valas, 1999]. The CDRs are the basis for billing of subscribers.

Furnish Charging Information (FCI) [ETSI, 1994] is an INAP signal that can be sent from an SCP to a SSP to make the exchange generate a TT. This is useful for all calls that do not generate TTs in the local exchanges. Examples are free phone and premium-rate calls. Those calls are examples of IN-services and the TTs are generated in the transit exchanges instead.

Settlement of accounts is done between network operators to be able to charge for network usage when one operator generates traffic in another operator's network. An example of this is when a call transits through a network, i.e. it neither terminates nor originates in the network. How this settlement of accounts is done is stated in bi-lateral contracts. The basic principle, though, is that the operator generating the traffic is charged. The exchanges that interconnect the networks of different operators keep track of transit traffic and generate charging data.

Chapter 3

Detaching Service Execution from Network

The telephone networks originally were developed for establishing calls between two parties. Nowadays the customers expect useful and sometimes complex services from the network infrastructure. The emergence of Internet and mobile phones has resulted in new types of service applications that make use of several different media or networks, i.e., cross media services. This, in turn, will force the telephone network operators to make use of, and offer cross media services.

3.1 Service Execution Structure

The role of the *Public Switched Telephone Network* (PSTN) has changed in the last years. In the beginning the only service provided by those networks was connecting two calling parties. The first step in the direction of designing more useful services was to introduce simple services like call forwarding. In the beginning each service had to be implemented in every switch in the network. Maintaining these services was not an easy task since it meant making simultaneous changes in all switches in the network. Therefore the services could differ from one area to another which was not desirable.

The solution to this problem was to centralise the execution of services in the network at *Service Control Points* (SCP). This is illustrated in figure 3.1 on page 14. Thus, the *Intelligent Network* (IN) was introduced and it was easier to update services and introduce new ones. However, with the IN solution the services are strongly connected and adapted to the IN platforms and therefore the lead times[1] are often long. Also see section 2.2 on this topic.

Lately the telephone networks have become a carrier of more complex services and by this change the telephone networks have come nearer to the packet switched networks. An example of a more complex service is a call forwarding service that is configurable via the Internet and routes a call to a certain destination depending on the time of the day or on the number of the caller.

Today there is also a trend among companies consuming telephony services to focus on the services offered via the network and the value they bring rather than

[1]The time from when a service is ordered until the service is delivered and operational.

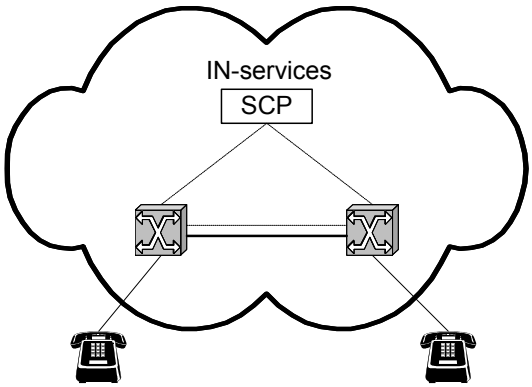


Figure 3.1: The centralised service architecture

focusing on owning the infrastructure. An example of this is that many companies subscribe to a virtual *Private Branch Exchange* (PBX) service instead of buying and installing the real hardware in the company office. Also, as stated in the example above, using the Internet in integrated services has become common. These cross media services encourage a complete separation of the service execution from the telephone network, and doing this is indeed, the tendency of today.

Consequently the two main service production structures of today are:

- *Centralised service execution* which means that the service is realised in the network domain. This is the traditional way of realising services in telephone networks. It provides limited flexibility since the service application runs in an closed environment inside the network domain and is not customisable from the outside. This structure will not be studied in more detail in this report.

- *Distributed service execution* which means that the service is realised outside the network domain. This solution is more flexible, but on the other hand it makes heavy demands on the integrity management of the network. See chapter 6. This is the focus for the reminder of this report.

3.1.1 Distributed Service Execution

The functionality of the telephone network is controlled by the signalling system. Thus services must interact with the signalling system to be able to make use of all network resources. As long as the service logic is located inside the network operator domain this will not introduce any problems because the interface to the signalling system is well known and the components interacting with the signalling system are trusted. This is the case with the IN and SCP solution.

To enable distributed service execution there has to be an interface through which network resources can be accessed by the service applications. Network resources could be methods for e.g. routing, authentication and billing. This interface must have well defined methods that the service applications can use

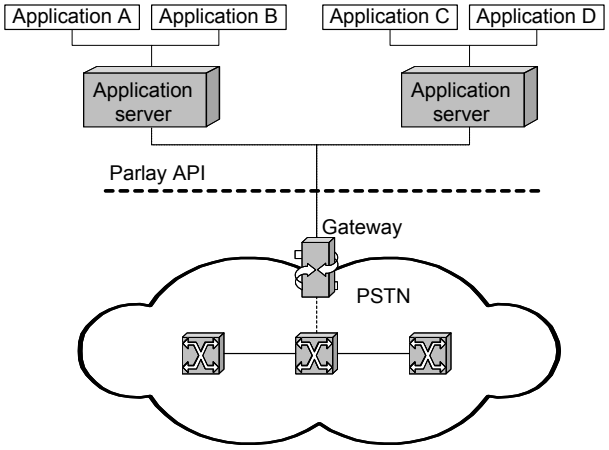


Figure 3.2: Distributed service execution

for triggering actions in the telephone network. However, the service can consequently be produced in an application that executes on a platform outside the network operator's domain. The service applications and its platforms are not always trusted. To have well defined methods for the communication between the service application and the network the interface could be implemented as an *Application Programming Interface* (API) on a gateway in the network (see section 4). Such an API can provide controlled access to the network so that the service application may use information from the network in critical decision-making such as call routing. The API has to be open and standardised to facilitate connection of service applications from different third party service providers. See figure 3.2.

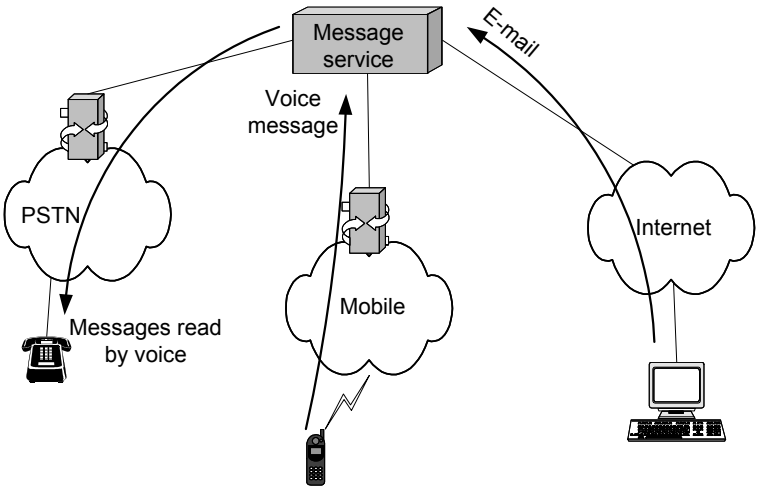


Figure 3.3: Multi access service

3.1.2 Multi Access Services

An important result of distributing service execution is the possibility to make a certain service accessible via several different media. One general example, that is illustrated in figure 3.3 on page 15, could be a message box where all kinds of messages (voice messages from fixed and mobile phone, e-mail, fax, and SMS) are stored. The user could access any message the way he wants. An e-mail could be read by a synthetic voice or a voice message could be sent as an e-mail, etc. The service, i.e. the message box, is exactly the same, but the access form differs depending on the users choice of media. This also means that all access networks taking part in the service must have some kind of gateway with an open and standardised interface through which the service application can control respective medium.

3.2 Driving Forces

This section will shed some light upon the driving forces behind the process of distributing the service execution.

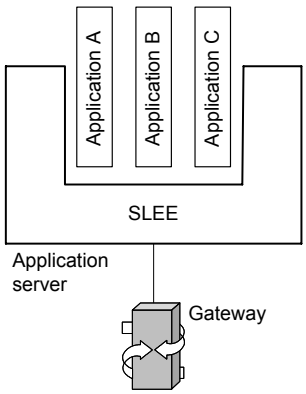


Figure 3.4: Service Logic Execution Environment, SLEE

3.2.1 Simplified Service Creation

The separation of service production from the transmission network domain does not only affect the service execution, it affects the whole life cycle of a service, i.e., design, construction, installation, delivery, operation, and maintenance. Moving the service execution outside the network is therefore a step towards shorter lead time, greater flexibility, and adaptability when developing and deploying new services in the network. Via the standardised interface, freestanding service providers may develop new and interesting services that easily can be distributed to the subscribers via the telephone network. Today, some companies[2] are offering so-called application servers that can host several applications that

[2]Examples are Incomit (www.incomit.se) and Appium (www.appium.com).

execute different services. These servers often have a *Service Logic Execution Environment* (SLEE)[3] based architecture, that include already developed tools and packages. This is shown in figure 3.4 on page 16. The service application development time therefore becomes quicker with lead times of months instead of years.

3.2.2 Increased Network Value

A network with moderately high traffic load is profitable and that is what the network operator wants. Distributed service execution opens up the network which leads to a simplified service creation process. Almost anyone with a good idea of an useful service could implement it in a standard programming language. This results in more services being added to the network. This includes both large scale services and small niche services. By increasing the number of useful services that are accessible via the telephone network the traffic and the value of the network will increase.

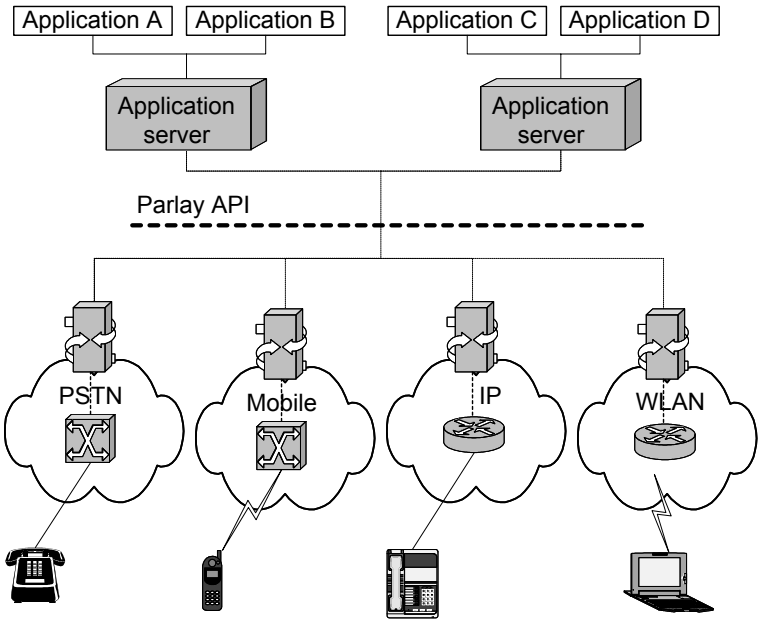


Figure 3.5: Convergence of services

3.2.3 Convergence

Another value driver for the service separation is convergence. Since the service execution platform, the *Application Server* (AS), is completely separated from the network it can serve more than one network as described in section 3.1.2. This will lead to convergence between different networks. Even networks of different types (e.g. IP networks and PSTN networks) can converge since the same

[3]Please confer JAIN SLEE at http://jcp.org/aboutJava/communityprocess/first/jsr022/

services can be offered in the two networks. For example, both networks may offer some kind of Internet configurable call forwarding. This means that the PSTN network still can compete with e.g. IP telephony with respect to services. Please refer to figure 3.5 on page 17. In addition, the services deployed on these network independent platforms can offer interesting and useful services that combine features from several different networks. Also the transition of subscribers from PSTN to IP telephony becomes smoother when the same services are connected to, and thus available to both networks. This is especially interesting since IP telephony will likely replace the circuit switched telephony eventually.

3.3 Endangered Network Integrity

When the service execution is separated from the access medium the integrity of the network may be harmed since some parts of the control of the network are given to a possibly non-trusted service application provider via the API. This is a "necessary evil" since some control has to be given to enable the service provider to produce and execute services, but at the same time the network operator must prevent the service provider from jeopardising the operation of the network. For a complete definition and analysis of the integrity problem, please see section 6.

Chapter 4

Parlay API

This chapter contains descriptions of the Parlay framework and service APIs and general concepts associated with them. The aim is to introduce different concepts and techniques to help the reader understand the discussions of different topics later in the report.

4.1 The Progress of Parlay API

The Parlay Group [Parlay Group, 2002] appeared in 1998 when the companies British Telecom, Microsoft, Nortel Networks, Siemens, and Ulticom joined their forces. Their aim was to define a set of *Application Programming Interfaces* (APIs) that support applications outside the secure network operator domain. The initial development was focused on call control, messaging, and security.

Until then the network operators jointly with network equipment manufacturers had designed, developed, deployed, and administrated the service applications in the network. Managing services was an inflexible and expensive process and therefore short-lived or niche service applications were considered commercially unfeasible by the network operators. Thus, the Parlay APIs opened up the network for third party service providers outside the domain of the network operator. By giving access to core network capabilities the development and deployment of new service applications on the network is facilitated.

The Parlay Group has expanded and today there are 19 companies that are full members. They regularly publish the technology independent specifications that define the set of interfaces constituting the Parlay APIs, i.e., the methods, events, parameters and their semantics. At the time of the writing of this report the latest Parlay version is 3.2, but in this report we describe the Parlay version 3.0 as this is the version we have analysed.

4.2 Distinctive Features

Parlay is an open *Application Programming Interface* (API) to telecommunication protocols such as INAP, ISUP, and SIP [Handley et al., 1999]. It facilitates an easy-to-learn interface that has schemes for naming, security, etc. The API is specified with the *Unified Modelling Language* (UML) [UML, 2002]. This means

that the API not bound to any particular programming language or architecture. The specification is also open and is available from the Parlay Group [Parlay Group, 2002].

The Parlay API abstracts network resources. Therefore there has to be a Parlay gateway translating Parlay API method calls into commands understood by the network. Please see chapter 3 for more on this topic.

The most important and distinctive features of the Parlay API are illustrated in the list below [Appium, 2002]:

- The network resources are independent. This means that resources available to a third party application via the API and the gateway do not need be part of the same underlying network or domain. Resources may very well derive from several different networks or any other type of suitable infrastructure.
- Secure access of third parties guarantees that no third party application connects to and gains access to the network (or other) resources without authorisation.
- Distributed architecture implies that the applications and the resources can reside in physically separated parts. The API could be used with any common technique for distributed systems such as CORBA [Corba, 2002] or RMI [RMI, 2002].
- The design is object oriented which makes it suitable for connecting applications based on object oriented languages such as Java [Java, 2002] or C++ [CPP, 2002].
- The API is manageable which means that the use of network resources can be limited by the network operator. An example of this is so called *Service Properties*. Please see section 8.2.4.
- The API is extensible. The object oriented architecture makes it easy to add and remove resources. This is facilitated via a discovery functionality. The third party applications are informed about the network resources[1] offered via the Parlay gateway.

4.3 The Parlay API Architecture

This section will describe the logical architecture of Parlay. The most important entities and their interfaces will be illustrated.

4.3.1 Overview

An API is a set of rules for writing subroutine calls that access functions. Programs that use these rules or functions in their API calls can communicate with each other using the API.

The applications deployed on third party *Application Servers* (AS), use resources defined in Parlay. Those resources are offered by *Service Capability*

[1] The network resources are called *services* in the specification of the Parlay API.

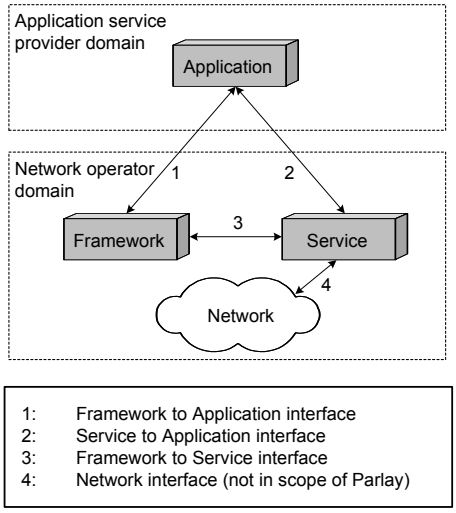


Figure 4.1: The Parlay API Architecture

Servers (SCS)[2] and provided to applications via the Parlay API on the Parlay gateway. The SCS implements the server side and the application implements the client side of the API. The SCSs interact with core network resources such as *Service Switching Points* (SSP), exchanges, etc. The resources themselves, provided to the applications via SCSs, are called *Service Capability Features* (SCF).

The Parlay API defines several categories of object-oriented interfaces on both the network side and the client application side of the API. These interfaces are illustrated in figure 4.1 on page 21. Each interface consists of a number of methods and they have several parameters that can be set. On the client application, the interfaces are call-back methods that are called from the network during a Parlay session. The strength of the API approach (for example, over database lookup interfaces) is that by defining a secure real-time interface there is a distinct boundary between the network operator and the third party application provider [Beddus et al., 2000].

Parlay is a very broad API covering a variety area of resources and services. This could be a drawback since many of the included APIs have never been realised in practice, i.e. there are no implementations. That is especially the situation for the second half of the APIs in section 4.3.4. Therefore, of all interfaces provided in the Parlay API we have centred our attention on the two most important categories since they are relevant to this report. That is the *Parlay framework API* and the *Call control API*.

4.3.2 The Framework API

The Parlay framework API provides the requisite surrounding capabilities for the Parlay service APIs to be open, resilient, secure, and manageable. The Parlay framework provides functionality that is common to all categories of SCFs, i.e., independent of the service type.

[2]This is a term used in the specification of Parlay API.

Interfaces between Application and Framework

The following parts form the API between the application and the framework.

Authentication
The application must be authenticated before it is allowed to use any of the other interfaces of the Parlay API and until this is done, any application-initiated invocation of an operation will fail. The application must have a reference to the Parlay framework it wants to access. This reference could be gained through an object reference in form of a string, a URL, etc. The reference is used to initiate the authentication process.

Authentication process is based on a peer-to-peer model. Cryptographic processes and digital signatures may support the authentication that need not be mutual. Thus, it is mandatory that the Parlay framework authenticates the application, but it is optional for the application to authenticate the Parlay framework.

At any time during the Parlay session, either side can request a re-authorisation (that need not be mutual).

Authorisation
The authorisation determines what a previously authenticated application is allowed to do, e.g., which SCFs it may access. In order to use a Service Capability Feature the application must establish and digitally sign a *Service Level Agreement* (SLA).

Discovery of Service Capability Features
The discovery interface may be used at any time after successful authentication. An application that wants to discover new SCFs uses the discovery interface in a three-step process:

1. Determine all service types that are supported by the Parlay framework
2. Determine the properties of a specified service type
3. Obtain a list of all SCFs of a specified service type and fulfilling the specified properties

Note that after authorisation from the Parlay framework the applications access the SCFs directly via references gained from the Parlay framework.

Establishment of Service Level Agreement
In this phase the conditions under which an application is allowed to use network SCFs are established. A SLA is a business level transaction where the *Application Service Provider* (ASP) agrees upon terms of use of a SCF with the Parlay framework provider. SLAs can be reached using either on-line or off-line mechanisms. The application has to sign the on-line part of the Service level agreement before it can access any SCF. The off-line part may be a exchange of control documents and input of parameters via a management system.

Access to Network Service Capability Features
The specified terms of use (security level, domain, etc) must be complied and therefore the Parlay framework must provide access control functions to authorise the access of SCFs from any application.

Event Notification
This interface is used to notify an application if a generic service event has occurred, i.e. registration of a new SCF in the Parlay framework.

Integrity Management
The integrity management interfaces include mechanisms for load balancing, fault management, and heartbeat.

- Load balancing between applications is supported according to a load management policy. It is possible to specify the load balancing rules the Parlay framework should follow for a specific application. This is related to the *Quality of Service* (QoS) level the application has subscribed to.

- Fault management is a mechanism for informing the concerned applications that a SCF has failed and no longer is available.

- Heartbeat allows the Parlay framework to supervise applications by requesting them to send out heartbeats at a specified interval. If the heartbeat is not received from the application within the interval the application has failed. This information might be used by the Parlay framework to perform some recovery measure.

Interfaces between Parlay Framework and Service Capability Server

Registration of Service Capability Features
All SCFs have to register in the Parlay framework to be accessible to applications via the discovery interface (see section 4.3.2). The SCFs are registered against a certain service type and the Parlay framework maintains a repository with the service types and registered SCFs.

Upon registration, the supplier of SCFs must provide some property values and a service type describing the SCF. Via those values and service types, the application may obtain lists with the SCFs it wants to use.

Interfaces between Parlay Framework and Enterprise Operator

This interface provides tools for realising a business model where the enterprise operators act in the role of subscriber or customer of services and the client applications act in the role of users or consumers of services. The framework itself acts as a retailer of services. However, this is mostly an administrative interface for handling business relations, and thus is not in scope of this report.

4.3.3 The Call Control API

The Call Control API consists of four different interfaces with different purposes. They are described in the following four sections. To explain the API we start by describing the Call Control model that contains four basic objects:

- *the call object*
  The call object is a relation between a number of parties and it is used to establish a relation between a number of parties by creating a leg for each party within the call. From the view point of the application, the call object relates to the whole call. Thus, when invoking a release method on a call object all associated parties and physical calls are released.

- *the call leg object*
  A call leg object represents the logical association between a call and an address. A call leg can be attached to, or detached from a call. When a leg is attached, it is connected to the other legs that are attached to the same call. This means that the media or bearer channels are connected and the attached legs can "speak" to each other.

  There are two ways for an application to gain control over a leg. Firstly, the application can request that it be notified when a call meets certain criteria and then the call can control the legs associated to it. Secondly, the application can create a call and consequently control the call and its legs.

  A leg object can exist without being associated with an address and is then considered idle. The leg object becomes active when it is routed to an address.

- *the address*
  The address is the logical representation of a party in a call.

- *the terminal*
  A terminal is the signalling end point for a party.

Generic Call Control

Generic Call Control is the basic control facility for the *Call Control API*. The facility is a bit blunt, but it contains important methods for handling simple call scenarios. For example, it does not give explicit access to the legs and the media channels. The number of legs in a generic call is also limited to two; one incoming and one outgoing. Multileg-calls are handled via the *Multiparty Call Control API*.

Multiparty Call Control

The *Multiparty Call Control API* extends Generic Call Control with the ability to manage individual legs. This means that several legs may simultaneously be attached to the same call. However, Multiparty Call Control requires INAP CS2 implementation in the PSTN, see section 2.1.4 on page 8.

Multimedia Call Control

The *Multimedia Call Control*, in turn, extends the functionality of the Multiparty Call Control by adding multimedia capabilities. Thus the concept of a media stream is introduced. These streams are generally negotiated between the terminals in a call. The media streams are bi-directional media channels that are associated with a call leg. In a multimedia call, the Multimedia Call Control can give control over associated media streams to the applications in the following ways:

- Multimedia calls with media streams that meet certain criteria (defined by the application) can trigger the application.
- The application can monitor the establishment and release of media streams associated with an ongoing call.
- The application can allow or deny establishment of media streams.
- Established media streams associated with a certain call leg can be requested and explicitly closed by the application.

Conference Call Control

The Conference Call Control inherits its generic properties from the Multimedia Call Control. It gives the application the ability to arrange conference calls with subconferences. Subconferences define groupings of legs within the main conference call. Only parties that belong to the same subconference have connections to each other and can talk.

When a conference call is initiated the application can create, split, or merge subconferences. The conference call must always contain at least one subconference. The application can move call legs between subconferences and retrieve a list with active subconferences within a conference call.

A Conference Call Control also supports the applications with resource reservation management. The resources are network dependent, but represent transmission capacity. It is possible for the application to reserve conference resources during predetermined time periods, free reserved resources, and search for available resources based upon certain criteria.

4.3.4 The Niche APIs

The following interfaces are specialised to certain areas of usage. Their detailed characteristics are not in the scope of this report but to do the API description complete we will give brief descriptions.

User Interaction

There are two interfaces defined for User Interaction:

- The Generic User Interaction is used by applications to interact with end users, i.e. announcement messages and to collect information from a user.
- Call User Interaction is an enhancement of the Generic User Interaction that is used to interact (collect and send information or messages) with parties that are connected via a call leg. Call User Interaction provides additional methods for recording and deleting voice messages.

User Interaction can be performed on a Call, Multiparty Call, or a call leg object and can involve one or more parties.

Mobility

Mobility service API contains functionality to support applications that involve mobility. The API is divided into four parts:

- *User Location*
  The applications can use this interface to obtain the geographical locations (co-ordinates) of users connected to a fixed, mobile, or IP-based telephone system.
- *User Location Camel*
  This interface is used by an application to retrieve the location-related information about the network, e.g. identification of a cell in a mobile-telephone network.
- *User Location Emergency*
  If an application is designed for handling of emergency calls, it can automatically retrieve the location of the caller via this interface.
- *User Status* This interface supplies an application with the status of specified users in fixed, mobile, and IP-based telephone systems. The provided status is of type user reachable, not reachable, or busy.

Terminal Capabilities

The Terminal Capabilities interface retrieves the latest available capabilities of a certain terminal. The provided information could be, e.g., terminal attributes and values.

Data Session Control

A terminal may, via Data Session Control request a data session and the associated application can reject or approve its establishment. The application can also continue the establishment, but change the destination for the data session. The Data Session Control consists of two parts:

- Data Session Manager contains functions for enabling or disabling data session-related event notifications.
- A Data Session provides the means to establish, release, and supervise data sessions.

Generic Messaging

An application can use Generic Messaging to send, receive, and store messages such as voice and e-mail. The messaging system is assumed to contain mailboxes, folders, and messages. Generic Messaging provides methods for handling such a system. It is a simple API that is not suitable for accessing Internet mail servers [Appium, 2002].

Connectivity Manager

The Connectivity Manager provides methods for an *Application Service Provider* (ASP) to establish Quality of Service parameters for packets travelling through the provider's core network. This API has its focus on IP-networks and is not directly applicable to PSTN.

Account Manager

The Account Management interface provides functions to applications for handling end user accounts. This is useful for applications that have to query account balances, account history, etc. and be notified about charge-related events.

Policy Management

The Policy Management interface addresses the creation, modification, and viewing of policy information. It also handles subscription to, and generation of events and handling of SLAs. SLAs may be used to convey authorisation for access or subscription to policy information or to modify or create policy information.

For an thorough exposition of the Policy Management concept, please see section 8.3. The Policy Management interface is not included in Parlay version 3.0. It is under development by the Parlay Group and included in version 3.1, hence it is a draft. The reason it is included here is that the policy management concept is part of our solution in chapter 8 and it is expedient for the reader to know that this API exists.

Charging

The Charging interface provides methods for applications to charge end users for their use of a specific application or data. A typical service is pre-paid calls.

Chapter 5

Parlay Gateway Operation

In order to use the Parlay API the network operator has to implement a Parlay Gateway. This gateway translates the calls of Parlay API methods into low-level operations that are understood by the signalling system in the underlying network. From the network point of view the gateway acts as an *Service Control Point* (SCP) and the signalling system is accessed via an *Service Switching Point* (SSP). This implies that the gateway is connected to the signalling system via a SSP.

5.1 Tasks of a Parlay Gateway

The Parlay gateway has several important tasks to perform. The most important though, is protocol translation and to supervise the network integrity.

5.1.1 Protocol Translation

The Parlay API (see chapter 4) specifies what API calls are available and the impact they will have on the network. In most cases the API calls are translated to INAP commands. However, the Parlay specification does not specifiy *how* this translation is made or to what low-level protocol in the signalling system. Those issues are implementation specific.

5.1.2 Preserving Network Integrity

Giving away control over the network via a gateway in a manner as described in chapter 3 leads to the integrity being put at risk to some extent. This integrity reduction has to be compensated for in some way. The easiest way to address an integrity problem of this kind would be to ensure that every application implementing a service is secure and not able to violate any rule protecting the network integrity. However, this is not a reasonable solution since according to the model of "third party service application provision" the applications are not under direct control of the network operator and the network operator can't possibly scrutinise and approve all connected applications. This would be a commitment reaching too far.

Instead there has to be some functionality in the gateway supervising the activity of the service applications and preventing them from committing integrity violating activities.

There are two categories of activities that are of utter importance and need be supervisioned by the gateway:

- Utilisation of network resources
- Charging for resource utilisation

5.2 Examples of Service Application Operation

This section will describe how the Parlay gateway inter-operates with service applications and the network, and what happens in the Parlay gateway when a service is executed. To do this, we will present two typical services below.

The information flow in the gateway is similar no matter what service application is using it. Some service applications requires multiple interactions between the network and the application through the gateway. Interaction may differ due to the initiative for a call coming from the network or from an application. A simple network initiated number translation would be a common usage of the gateway. The interacting components in the gateway environment are illustrated in figure 5.1.

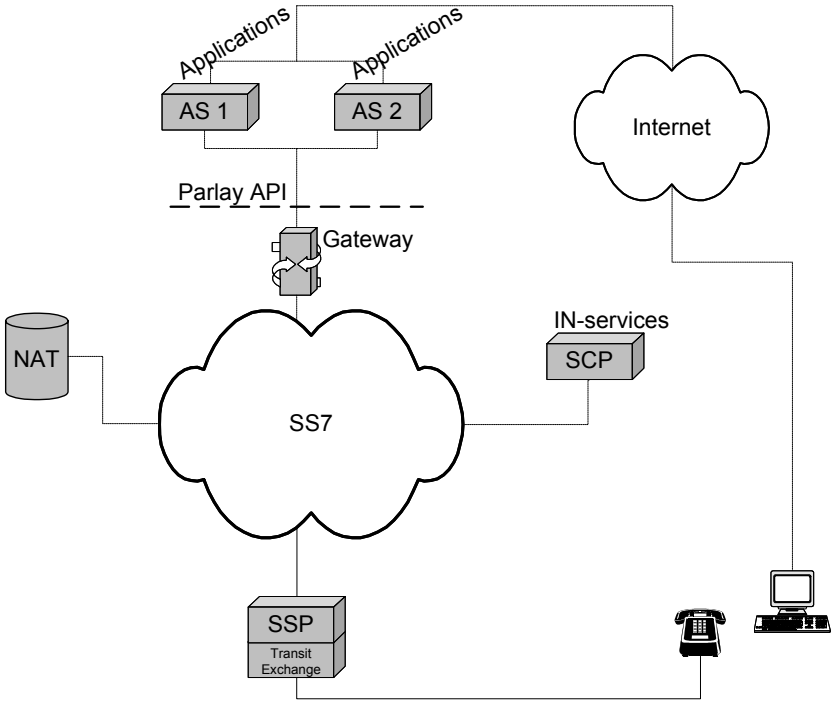


Figure 5.1: The Parlay gateway and interacting components

5.2.1 The Click-to-dial Service

The first example is a Parlay application on an *Application Server* (AS) that enables call set-up via an address book. To call a person the user clicks on a name in an address book. The service application then sets up the call.

Usage Scenario

1. The user (caller) chooses a person to call in the address book
2. The application creates a call and connects it to the user's own predefined telephone
3. The user answers
4. The application connects the call to the chosen number
5. The callee answers
6. Call in progress
7. The call ends

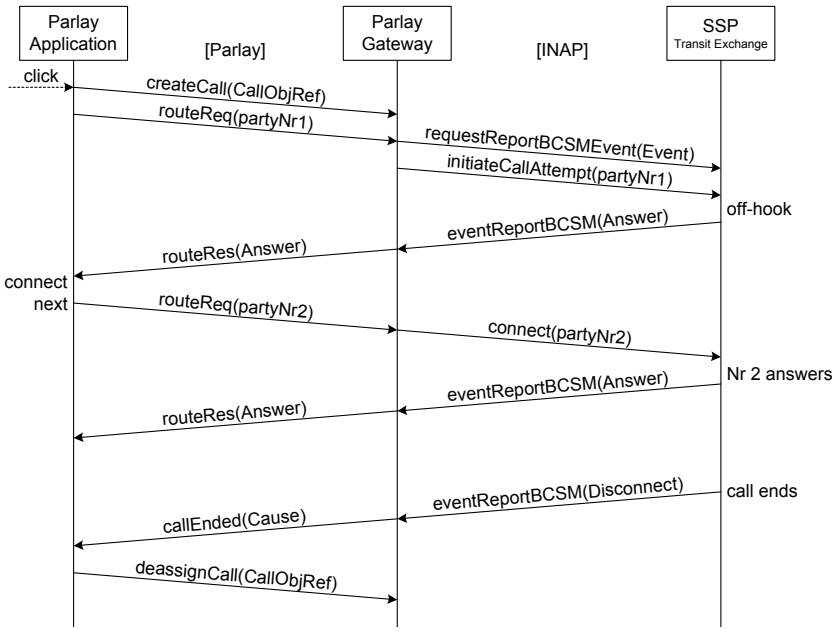


Figure 5.2: Click-to-dial sequence diagram

Sequential Description

This section will describe how the *click-to-dial* service is executed chronologically according to figure 5.2 on page 31.

The *click-to-dial* service application is informed when the user clicks on the button to dial the chosen person. It receives A and B numbers[1] from the computer terminal of the user. The terminal could be connected to the AS via the Internet. The application then calls the method `createCall()` on the gateway via the Parlay API. This call is handled by a *Call Control Manager* that creates a call, i.e. instantiates a call object that is responsible for this specific call in the gateway. The application then calls the method `routeReq()` on the gateway call-object to set-up a call-leg to the A number.

During the initiation phase of the *click-to-dial* service application it has requested that it be notified of events concerning its calls (answer, disconnect, no answer, busy, etc.). The gateway, in turn requests the network to give it information or notification about the call by issuing the INAP `requestReportBCSMEvent()` signal to the SSP. The gateway will get notifications about all specified events until the call has ended. This is vital since the application needs to know immediately when the caller has answered. Consequently, an SSP is instructed (by the gateway) to notify the gateway when the caller answers and then it requests the gateway to set-up a call leg to the A number by issuing the INAP `initiateCallAttempt()` signal.

The call leg is set-up like any other call by ISUP signalling (see section 2.1.4 on page 7) between the transit and local exchanges in the transmission network and will not be dealt with further here.

The SSP notifies the gateway when the caller answers via the INAP `eventReportBCSM()` signal. If the A number is busy or non-existent, then this information would be sent to the gateway in similar way.

The gateway passes this notification on to the application by calling the Parlay API method `routeRes()`, i.e. that is the result of the first `routeReq()`. The application calls the Parlay API method `routeReq()` again on the same call object in the gateway to set-up a call to the callee. This time with the number of the callee as the argument.

Subsequently, the gateway issues the INAP `connect()` signal with the number of the B-party as argument. This implies that the transit exchange sets up the last outgoing leg to B and connects it with the first leg. The call is established in the network, i.e. the two participating parties are connected.

When the B-party answers the network sends a new INAP `eventReportBCSM()` signal. This signal is passed on to the application by calling the Parlay API method `routeRes()`.

Finally, the call ends and the SSP is informed via an INAP `eventReportBCSM()` signal. The gateway then notifies the application via calling the Parlay API `callEnded()` method. The application is then expected to call the Parlay API `deassignCall()` method. This releases the Parlay call-object in the gateway, but leaves any associated ongoing calls in progress.

[1] A is designating the caller, and B is the designation of the callee. This notation is conventional when describing calling scenarios.

5.2.2 Web Configured Call Forwarding Service

The second example is a service where the user may configure supplementary services via a web page.

Usage Scenario

1. The user states the conditions for call forwarding on a web page (forwarding numbers, time periods, etc.)
2. An incoming call triggers the application which calculates the new destination number depending on the present circumstances (calling number, time, etc.)
3. The application connects the call to the new number
4. The called person answers
5. Call in progress
6. The call ends

Sequential Description

This section will describe how the *web controlled call forwarding* service is executed chronologically. This is illustrated in figure 5.3 on page 34 and in figure 7.1 on page 45.

First the exchanges in the network are configured to handle the number, to which the service is connected, as an *Intelligent Network* (IN) service. This is done via some management system and is not in the scope of this description. The consequence of this configuration is that when someone calls the number of the user, the decision of where to connect the call is handed over to an SSP on a transit exchange. In turn, the SSP know that it should ask the gateway where to connect the call.

The application informs the gateway about what numbers the application wants to handle by calling the Parlay API method `enableCallNotification()` with arguments specifying triggering criteria and a reference to the *Call Control Manager*[2] in the application. The event criteria specifies what events to trigger on. An example could be the way a certain number has been triggered[3].

When the caller makes a call, an INAP `initialDP()` signal is issued from the network to the gateway. This signal includes information about calling party, called party, call forwarding, etc. This signal is transformed into a call of the Parlay API method `callEventNotify()` at the application that analyses the event and calculates a new destination for the incoming call.

The application then calls the Parlay API method `setCallback()` to create a reference to the application call object that is in control of the call. Subsequently the application calls the method `routeReq()` with the number of the

[2]This Parlay object residing in the service application instantiates and handles call objects.

[3]There are several events that may trigger an application. For example a number could be triggered in two different ways: an subscription number could be called (terminating) or it could make a call (originating). In both cases the same subscription number is involved, but the events are different.

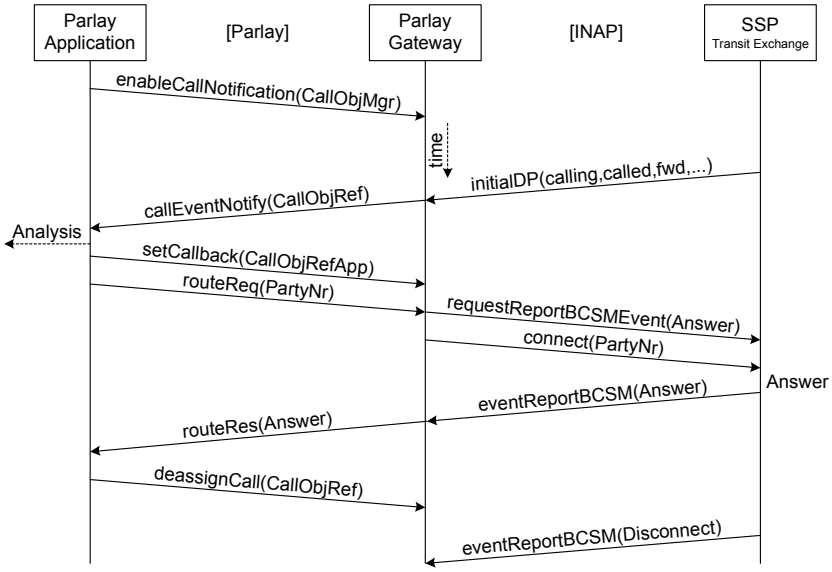


Figure 5.3: Web configured call forwarding sequence diagram

callee as argument, i.e. the new destination for the incoming call that triggered the application.

To propagate the new destination into the SSP in the network the gateway sends two INAP signals. The first one is `requestReportBCSMEvent()` specifying what events the gateway is interested in regarding the present call. Events of interest include: if the new destination is e.g. busy, answering or not answering. The gateway will get notifications about all specified events until the call has ended. Subsequently, the gateway instructs the SSP to set-up the establish the call by issuing the INAP `connect()` signal.

When the callee answers the gateway is notified via the INAP `eventReportBCSM()` signal from the SSP. This information is transfered to the application via a call of the Parlay API method `routeRes()` in the application. Now the application knows that the call is successful (i.e., in progress) and it therefore calls the Parlay API method `deassignCall()` with a reference to the call object in the gateway as argument. This action releases the call object in the gateway, but leaves the ongoing call itself in progress.

Finally when any of the calling parties hang up the SSP issues the INAP `eventReportBCSM()` signal. At this point this signal is of minor interest since there is no longer any call object associated with the call in the gateway.

Chapter 6

Network Integrity

The Public Switched Telephone Network (PSTN) is a vital part of the society. It has always been built to ensure high operational reliability. Many important information services such as emergency calls rely on this network. The network also provides an important improvement in the quality of life for lots of people by facilitating communication between people. All these users expect that the telephone network is available whenever they want to use it. To maintain the high availability and create a robust telephone system many measures have been taken, such as duplication of hardware, separate power supplies, creation of alternative routing paths, etc. All those measures contribute to the integrity of the network.

This chapter deals with the definition of network integrity by giving the conditions, describing the general components, and giving our adapted definition of network integrity.

6.1 The Necessity of Network Integrity

Network integrity is about designing a network that is reliable independently of the surrounding conditions. This issue has been accentuated because networks are becoming more and more complex, both in hardware and software. The more complex a system is, the more likely it is to fail because there is a large number of components that can break or subsystems that can fail in their co-operation.

A new aspect of network integrity issues has arisen in connection with the development of the third party network access. Regulatory initiatives and demands for new services have forced network operators to open up their networks (see section 3). The development and deployment of new services in the network require that third party service providers gain access to the signalling part of the network. They can issue commands to control calls and network resources in detail. This access is provided via a gateway implementing some *Application Programming Interface* (API). In the case of Skanova this is a Parlay gateway (see section 4). However, this introduces new integrity problems that need to be handled. For example, it is necessary to draw up contracts that regulate what the third party service provider is entitled to do in the network. Still the network operator has to guarantee the integrity of the network towards the customers. Thus the operator may be required to supervise the activity of the third party service provider and also take countermeasures if some activity is jeopardising the

network's integrity. The network operator preferably implements these surveillance and countermeasure functionalities in the gateway through which the third party service provider issues their control commands.

6.2 Definition of Network Integrity

Defining network integrity unambiguously is a hard task since this term is very broad. It ranges over several areas such as system architecture, behaviour, and functionality. The essence of this term however, is that the network acts according to the expectations and specifications of the owner of the network and that this behaviour is a steady-state. If we look in literature we find some general definitions. One example is found in Telecom Glossary 2000 [ANSI, 2001].

> "[System integrity is] that condition of a system wherein its mandated operational and technical parameters are within the prescribed limits."

This definition corresponds to the intuitive conception, but it does not emphasise the stability criterion. There is another definition [ANSI, 2001] that does:

> "[Network integrity is] the ability of a network to maintain or restore an acceptable level of performance during network failures by applying various restoration techniques, and mitigation or prevention of service outages from network failures by applying preventive techniques."

To make this definition operational it is necessary to assign the words "acceptable level", "various restoration techniques", and "preventive techniques" well defined values and meanings. The assigned properties are, of course, subject to different interpretations depending on which system or network the definition is applied.

The problem is that it is hard to get a tangible understanding of what network integrity is when talking in general terms. For this reason Ognjen Prnjat and Lionel Sacks have proposed a more detailed definition [Prnjat and Sacks, 2000, Prnjat and Sacks, 1999] including a number of attributes that are related to the integrity concept. Their definition is based on the following definition of system integrity by Keith Ward [Ward, 1995]:

> "Integrity is the ability of the system to retain its specified attributes in terms of performance and functionality."

The approach of including several attributes in the definition is beneficial because it makes it easy to apply the definition and test for any integrity violations.

6.2.1 Integrity Attributes

Ognjen Prnjat and Lionel Sacks list integrity attributes [Prnjat and Sacks, 2000, Prnjat and Sacks, 1999] below. In the following paragraphs these attributes will be explained. However, the proposed attributes are not absolute and other attributes may well supplement the definition of integrity:

- Availability
- Complexity
- Data coherence
- Feature interaction
- Liveness
- Performance
- Reliability
- Resilience
- Robustness
- Safety
- Scalability
- Security

Availability

Availability is the degree to which a system (e.g. network) or equipment is operational and conforms to its specifications at the start of a mission, when the mission is called for at an unknown time. The conditions determining operability must be specified so that it is decidable when the availability criterion is not fulfilled. If we express it mathematically, availability is the ratio of the total time a functional unit is capable of being used during a given interval, to the length of the interval. Availability is often given in percentage (e.g. 99.9997%).

The integrity of a system is of course jeopardised if the system has an availability that falls below some expectation or specification of the owner of the system.

Complexity

Complexity describes how demanding a certain task is to solve depending on the presented problem. The complexity of a system can be analysed on many levels. For simplicity we will focus on the following four areas:

- Computational complexity is the magnitude of time needed (proportional to the number of CPU cycles) to execute a task. This is related to how well a given procedure or algorithm can solve the problem.
- Data complexity is the magnitude of internal data (proportional to memory usage) needed to execute a task and this is related to what and how data structures are used and their interdependencies.
- Communication complexity is the magnitude of the number of messages that have to be issued to perform a certain action.
- Coupling is a kind of semi-complexity. It describes to what extent the system is connected and dependent on other systems. A high level of coupling indicates a high level of interdependence. Via coupling paths integrity violations can propagate between different systems.

If the system possesses a high level of complexity this might endanger the integrity of the system since high complexity makes heavy demands on the system operation. However, the complexity may also be used to increase robustness, security, etc. and therefore this is a question which needs careful weighing of the included attributes.

Complexity also influences the scalability of the system which is related to integrity. A high level of data complexity might result in an excessive memory usage leading to failure when expanding the system.

Data Coherence

Data coherence is an important aspect of control strategies in distributed systems where information is distributed over several locations and need remain consistent through time and change of circumstances.

Thus data is coherent if the value returned by a read operation to a shared memory location is the value of the last write operation to that location. If this is not the case and differences in information arise, then the integrity of the system is jeopardised. The incoherent information could then be interpreted in a way that it was not meant to and thus make the system fail.

Feature Interaction

In software development a feature is a component of additional functionality – additional to the core body of software. Typically, features are added incrementally, at various stages in the life cycle, usually by different developers. In a traditional telecommunications service, examples of features are a call forwarding capability, or ring back when free; a user is said to subscribe to a feature.

Features are usually developed and tested in isolation, or with a particular service. But when several features are added to a service, there may be interactions (i.e. behavioural modifications) between both the features offered within that service, as well as with features offered in another service. While particular interactions may be benign, in general, interactions can be severely damaging to system development and to user expectations [Calder et al., 2002].

As an example of feature interaction, the following can be considered. If a user who subscribes to call waiting and call forward when busy is engaged in a call, then what will happen when there is a further incoming call? If the call is forwarded, then the call waiting feature is clearly compromised, and vice versa. In either case, the users will not have their expectations met.

Operating different features or systems (with well defined and understood behaviour) together can consequently give rise to unexpected behaviour and this is incompatible with a high level of system integrity.

Liveness

A system may be described by its states. The system may have several states with different properties and the system always is in one (and only one) of these states. Each possible state describes the properties of the system and to which other states the system may change. State transitions occur as consequences of events within or outside the system.

The liveness property means that something “good” (or useful) eventually will happen in the system [Alpern and Schneider, 1984]. For the system to work properly it is required that the system may freely change states according to the specifications. If this is not the case the system will not work as predicted and neither is the system live. One of the two following conditions then apply:

- The system is in the state of *deadlock*. This means that the system is blocked in a state expecting a message or an event which cannot occur.
- The system is in the state of *livelock*. This means that the system oscillates between a closed set of states which it cannot leave. Despite ongoing state transitions no purposeful progress will be made.

Both these conditions are disadvantageous and they vastly reduce the possibilities of the system to provide its functionality. A system that has a high level of integrity accordingly has to preserve liveness.

Performance

Performance is related to the throughput of the system and is measured in number of successful transactions (e.g. sorting of data base records) per time unit. Performance could also be related to the response time, i.e. how long time it takes to execute a job.

A system that has a poor performance (few transactions per time unit or long job execution time) will also put its integrity at stake since there could be problems meeting system deadlines.

Reliability

The reliability of a system is the probability that the system, including all hardware, firmware, and software, will satisfactorily perform the task for which it was designed or intended, for a specified time and in a specified environment [ANSI, 2001].

A system that is expected to have a high level of integrity must be reliable. Reliability can be improved by redundancy i.e. the system has surplus capability and therefore it is unlikely to fail in its operation.

Resilience

Resilience is the ability of a system to recover from faults without outside help. This is an important property for networks if an link or a node goes down. In IP networks this behaviour is supported by routing protocols such as *Open Shortest Path First* (OSPF) [Stallings, 2000]. Every node in the network keep track of the state of the links connected to it. Resilience is infused into the network by every node transmitting its link state record to all nodes in vicinity. If a link goes down the traffic is spread over the remaining links.

The ability to recover from faults is an important property for integrity because it makes a system likely to continue its operation despite encountering failure.

Robustness

The term robustness indicates the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environment conditions [SEI, 2000]. Robustness and integrity walk hand in hand. The more robust a system is, the more likely it is to retain its integrity.

Safety

The safety property states that some "bad" (or unfavourable) thing does not happen during a procedure in the system [Alpern and Schneider, 1984]. This means that the system must not end up in a state in which it does not belong. An example of the safety property is keeping to the *first-come-first-serve* (FCFS) principle. It states that requests are served in the order they are made. A violation (a "bad" thing) of this safety property would then be serving a request that was made after one not yet served.

Violating the safety property implies that the system does not operate normally and this may jeopardise the integrity of the system.

Scalability

Scalability is defined as the ability of a system (hardware or software) to continue to function well (retain its performance, see page 39) as it, or its context, is changed in size or volume in order to meet user needs. In the main, the degree to which a system is scalable is determined by the architecture and the system complexity (see page 37).

A system that scales poorly puts integrity at stake when expanding as the operation of the system deteriorates.

Security

Security is a condition that results from the establishment and maintenance of protective measures that ensure a state of inviolability from hostile acts or influences [ANSI, 2001]. A system that is in a state of inviolability will also keep a high level of integrity since it is more likely to stay in its correct operational state and not able to be brought down by any unauthorised users.

Finally, the reader is kindly asked to observe that all of the mentioned attributes are interrelated and must be studied in the light of each other. An obvious example of this is a system with poor scalability that will decrease in performance as more entities are added to it. Loss of performance implies reduced availability and this may end up in having an negative effect on the data coherence and liveness due to timing problems between dependent systems.

6.2.2 Our Definition of Network Integrity

As the reader may have noticed network integrity is not an unambiguous conception. In this section we will summarise and give our definition. We will then look at the problem from the point of view of the network operator Skanova and not primarily from the point of view of the end users.

Again, Keith Ward states [Ward, 1995]:

> "Integrity is the ability of the system to retain its specified attributes in terms of performance and functionality."

It is in the interest of the network operator to have a well performing network with the all necessary functionality[1]. Consequently, our interpretation of the

[1] It is reasonable to presume that the network operator also has the customer's best interest in mind, otherwise he will start losing them to other operators.

quote above is that the "attributes in terms of performance and functionality" will be specified by the operator of the network. It follows that we, in addition to the attributes in section 6.2.1, will add the charging aspect to the requirements of network integrity. Being able to charge users for their use of network resources is an absolutely essential functionality in addition to the core network functionality used to create the services offered to the users. If you cannot charge then there is often no point in offering services.

So, this is our definition of network integrity:

> Network integrity is the ability of a network to steadily remain in a safe state, while performing according to the expectations and specifications of its operator, i.e. delivering the expected functionality and providing means to charge for utilised network resources.

Most integrity issues arise from the users of the network. Consequently, a network without users would be unlikely to end up in a state of poor integrity because there would be very little activity able to violate the integrity attributes. Integrity violations arise in consequence of the user's unintentional misuse or intentional abuse of the network resources or functions[2]. The network must stand these categories of trials.

To facilitate the interpretation of the attributes in section 6.2.1 we have translated them into two potentially dangerous cases. To avoid jeopardising the network's integrity, the network must be prevented from reaching a state where:

- call control related functionality conflicts with its specifications or
- the means for charging for resource utilisation is missing or erroneous

6.3 Factors Reducing Network Integrity

Different factors may have a negative effect on the integrity of the network. Faulty hardware could be one cause of reduced integrity. However, in this report we will not consider such integrity violations originating from defective hardware. Instead we will deal with integrity issues related to the functionality of the software that untrusted third party service providers connect to the network via the Parlay gateway.

In the following two sections we give representative examples of events and factors that may indicate that network integrity is at risk. For an exhaustive exposition of this topic, see chapter 7.

[2]There could be integrity violations even though the users make use of the network in a "correct" manner. However, this problem must be attributed to some apparent erroneous operation of the network. We assume the network operates according to its specification.

6.3.1 Call Control Related Functionality Conflicting with its Specifications

If network resources are utilised in a way not foreseen by the network operator there will be great integrity problems since the operator is not in control of the network and the behaviour of the network may be undefined. The following events may indicate integrity problems deriving from "bad" call control functionality:

- execution of illegitimate procedures
- repeated calls of demanding procedures
- misdirected calls
- pointless or nested call forwarding
- repeatedly unsuccessful call attempts
- excessive call set up

6.3.2 Lack of Charging

It is unprofitable for the network operator if the network were to end up in a state where no charging for resources was possible. In this case, the network operator would not get paid. The following events may indicate that there will be a lack of charging:

- insufficient information about a call or
- inability to transfer call information between systems

6.4 Enforcing Network Integrity

It is impossible to know how the third party service provider is going to make use of the network's resources. To be able to enforce a high level of integrity it is necessary for the Parlay gateway to be able to detect harmful behaviour of a third party service provider and take preventive measures when needed.

The Parlay gateway may also be involved in other measures that are not directly associated with the resource utilisation of the third party service provider. The Parlay gateway may need to assist the network with tasks that otherwise would not be done, but are essential for the integrity. An example is generating charging information when this is not done by any of the standard procedures.

Please note that it is not always possible to foresee or detect all kinds of threats against the integrity of the network. These issues are treated more exhaustively in chapters 7 and 8.

Chapter 7

Identifying Integrity Issues

The identification of integrity issues is a challenging task. Before launching an open service access system like the Parlay gateway, the network operator has to predict as many integrity risks as possible, and have a readiness and a strategy for new risks that will arise in the future. The introduction of new service applications will also introduce new problems. It is important for a network operator to take into account that the network might be used for different purposes, and users might start acting according to new patterns in the future. Thus, it must be possible to take action in a flexible way when new integrity risks arise.

In this chapter we will give some examples of typical integrity issues. These issues will give an idea of what to think of, and what different kinds of integrity risks one can expect to encounter. These aspects should be considered when designing a gateway. In chapter 6 we have defined what are to be considered as integrity risks. The definition of integrity can be divided into subsections, and according to the delimitations in section 1.4 we will give examples of problems regarding the areas *Call Control* (section 7.2.1), and *Charging* (section 7.2.2). In chapter 8 we are suggesting some methods for solving integrity problems.

7.1 Examples of Service Applications

To find some concrete integrity risks we have considered a number of existing and projected services. These include services implemented on existing AXE-exchanges and IN-platforms that possibly could be moved to application servers and executed via the Parlay gateway. We have also analysed the application server market, and services that are planned to be incorporated in the Skanova Parlay gateway project. This gives us a picture of what Parlay services are being deployed. During this study we have categorised the application services into three categories, which we will describe below. By looking at services in the different service categories, different problems and issues that need consideration will be illustrated. For each service category we will describe the characteristics.

In section 7.2 we will analyse what areas that are problematical and need be scrutinised. To have the reader better understand the integrity problems we will also give technical explanations.

7.1.1 Common Entry Services

Common entry services are open to several (all) users and are accessible via only one telephone number, i.e. the "entrance" to the service is common for all users. This type of service is probably one of the most widely used today in the PSTN network. Examples of such services are free phone services (access number beginning with 020 or 0800) and directory enquiries services (access number beginning with 118). Below we will describe a *Virtual Call Centre*, and a *Directory Enquiries Service*.

Virtual Call Centre

A *Virtual Call Centre* (VCC) service is a service that enables companies to allocate their call centre telephone operators at several different places. This means that some telephone operators could work at home and others could work in the office or wherever they happen to be as long as they have a telephone. Because they do not have to be in the same physical premises the call centre is *virtual*.

When a call is placed to a VCC service the service on the application server connects the caller to a feasible telephone operator. An example of a feasible telephone operator is an unoccupied operator in the same numbering area, since connecting to a far-off operator when someone nearer is unoccupied would imply a non-optimal utilisation of the network. A possible usage scenario:

1. Someone calls the virtual call centre
2. The application on the application server connects the call to a feasible telephone operator (this is a simple number translation)
3. The operator answers
4. Call in progress
5. Call ends

Directory Enquiries Service

A Parlay application listens for calls to a specific number, e.g. 118118[1]. When someone calls the service number, the application directs the calling party to an available telephone operator (a simple number translation). The telephone operator can give the calling party the enquired number orally, or connect the call to the enquired number. The caller should be charged for the follow-on-call, just as if the call where made as a separate call. A conceivable usage scenario:

1. A call is placed to a directory enquiries service, e.g. 118118
2. The application connects the call to a feasible telephone operator
3. Telephone operator answers

[1]This is the number to the Swedish directory enquiries service operated by Respons AB.

4. Call in progress
5. The call is connected to the enquired number (optional)
6. Call in progress
7. Call ends

7.1.2 Individual Entry Services

Individual Entry Services are services assigned for individual usage. The service is tied to a personal subscription or attached to a certain access number. One example of such a services is voice dialling[2], a service that is reachable via a subscriber specific number and used as a voice controlled address book for dialling orally. Other services are voice mail, cash card, supplementary services, blacklist and whitelist[3], etc. Voice mail is a widely used individual service in particular mobile network. We will here describe a *Web Controlled Call Forwarding Service*, and a *Blacklist Service*.

Web Controlled Call Forwarding Service

Services such as call forwarding, completion of calls to busy subscribers, and three party calls are examples of supplementary services. Today they are controlled via the * and # buttons on the telephone. If these services were possible to manage via the web instead a lot would be gained in the area of user-friendliness. For example, one could have a profile that changes during the day. This service is illustrated in figure 7.1.

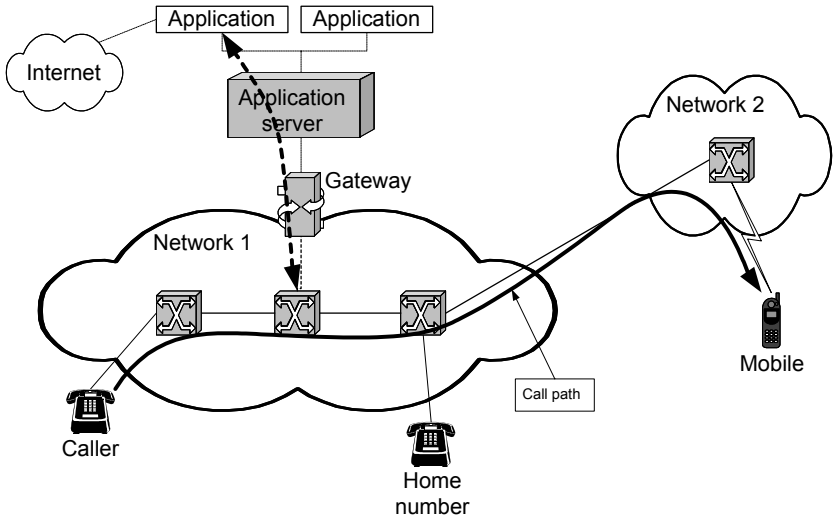


Figure 7.1: Web configurable call forwarding service

[2]An example is the service "Telia röstrigning".
[3]A blacklist defines numbers that should be rejected, and a whitelist defines (the only) numbers that should be accepted.

The supplementary service can also easily be combined with a so-called personal number[4]. A personal number is not tied to a physical subscriber line access, instead the user has to decide where calls to a personal number is to be forwarded.

From the view of the network, this kind of service is a simple number translation. Since the applications on an application server, easily could be connected to the Internet, the service could be web controlled. This gives the user the opportunity to pre-program call forwarding that is dependent on several circumstances. A possible usage scenario:

1. A call is directed to someone's home number, e.g. 08-234 234, or to a personal number, e.g. 075-123 123
2. The application on the application server computes where to forward the call and subsequently connects the call, e.g. to a mobile phone
3. The callee answers on his mobile
4. Call in progress
5. Call ends

Blacklist

A blacklist service prohibits calls to be received from some predefined numbers, or ranges of numbers. It can be implemented in different ways, e.g. the caller can get no answer, or a busy tone when calling to the subscriber from whom he is blacklisted, and the telephone at the callee does not ring. A feature could be a list of call attempts from blacklisted numbers on the configuration web page of the service.

7.1.3 Application Initiated Services

Application Initiated Services is a category where all services have in common that the initiative to set-up a call comes from an application service. This means that there is no trigger in the network initiating the service and the service is not accessible by any access number. Examples of such services are wake up calls, advertisement calls, etc. A *Click-to-Dial* service is described below.

Click-to-Dial

This is the umbrella term for applications that set up calls after being initiated via e.g. a web page. Thus one can make a call by clicking on the web page instead of dialling the telephone. One example of such a service is if the user's address book in a computer is equipped with the function "call" in addition to

[4]In Sweden such a personal number has prefix 075.

"send mail". Another example could be if advertisements on the web have a field where it is possible to enter your own number and then click "dial" to call the advertising company. In the first case the caller pays, in the second case the called advertising company pays for the call.

7.2 Integrity Risks Related to Functionality

The services described above can, if no action is taken, give rise to a number of integrity problems. Some issues are common to all services, and some are specific to a particular service and event. All issues can more or less be classified into different categories according to which aspect of integrity they violate. We will give some examples of integrity issues that illustrate different types of integrity problems. The integrity issues are, as said in the introduction, related to functional aspects of the services and the network.

7.2.1 Call Control

The integrity issues in this section are all related to call control. When new services, that are able to control calls, are introduced in the network, caution must be taken to protect the network from faulty behaviour. The following items will illustrate this.

Access Control

The first problem to look into is the access to the telephone network and signalling system, which goes through the gateway. Only authorised application servers and applications, should be allowed to access the gateway. Applications should only be allowed to access predefined features or resources in the network.

- The integrity is jeopardised if there is no control over which applications connect to the gateway.

Destination Number

For some services the gateway need have control over which subscriber numbers calls are connected to, since there may exist numbers to which it is forbidden to forward calls. For supplementary services there are some numbers that should be impossible to forward calls to, e.g. emergency numbers.

- The integrity may be harmed if there is no control over where calls are connected.

Feature Interaction

Some services in combination with each other can give rise to integrity issues. One such case is the combination of the services *blacklist* and *call forwarding*. The network integrity may be jeopardised if the following take place:

1. A has B on his blacklist, i.e. B cannot call A.
2. C has forwarded his telephone to A.
3. B calls C and the call is forwarded to A.
4. Since C is not on A's blacklist, the call gets through to A.

This erroneous behaviour could be prevented if the blacklist based its decisions on numbers the call has been redirected via, as well.

- The integrity may be harmed when services interact in an unexpected way resulting in undesired events.

Number of Origin

Many applications might like to receive the calling party number, some applications are even based on that information. A problem might then arise when someone with an unlisted[5] telephone number uses this service. The callers unlisted number must not, under any circumstance, be shown to the callee. If so, the integrity is harmed. A third party service application is regarded as untrusted, since the network operator cannot be sure of what the service provider does with the information. The number could be presented, on e.g. a web page or an invoice.

If a caller with an unlisted number calls a call centre, the closest telephone operator cannot be found based on the callers number. This restriction can be problematical for services that depend on the location of the caller, e.g. traffic information or assistance services. One solution is to regard some applications as trusted, and allow unlisted numbers to be shown to trusted applications. In the gateway one would then like to have the opportunity to decide if unlisted numbers should be passed to an application or not, depending on it's trustworthiness.

- When restricted information is illicitly spread to unauthorised part, the integrity is harmed.

Notifications

Applications cannot be allowed to request and receive just any kind of information from the network. In theory an application could subscribe to "off-hook" events for every telephone in the network. The gateway would then be overloaded with event notifications from the network. It is necessary to be able to restrict what kind of information an application should be able to receive, trigger on, etc. This restriction should be possible to apply to all applications connected to the gateway.

- There is a threat to the integrity if the gateway runs a risk of becoming overloaded.

[5] Unlisted number implies that the subscriber number is not listed in the telephone directory book or equivalent, and consequently must not be distributed to a third party.

Loops and Livelock

Loops can appear both in the signal and the transmission part of a telecommunications network. Loops in the transmission network are handled by the switches in the network itself and need no special attention in the gateway. The SCCP protocol in SS7 has functionality for counting the circuits reserved by a call [Modarressi and Skoog, 1990]. In the Skanova network one call is allowed to occupy five circuits. This could happen if subscribers are doing several successive call forwardings.

By loops in the signalling system we mean for example two SCPs continuously referring a call to each other. I.e. an SSP asks the SCP about where to connect a certain call. The answer is that the SSP must ask another SCP, which redirects the call to the first SCP[6]. This will lead to a lot of traffic in the signalling network but no real loops. However, the call will not reach any destination. This scenario can for example appear if two users with the supplementary service *call forwarding* redirects their phones to each other. If the fields in the INAP operation `Connect()` (such as `originalCalledPartyID` and `redirectingPartyID`) are filled in correctly, the exchanges in the network should be able to detect these loops and prohibit them.

Loops including the NAT[7], see section 2.3, could appear to customers with supplementary service and personal numbers (numbers with prefix 075). Setting the *Nature of Address* (NoA) to an appropriate value can prohibit these loops. The NoA is an indicator sent along with the INAP `connect()` operation from the gateway. The NoA can be set to indicate that the number is already checked against the NAT. This prevents the network from issuing new requests to the NAT regarding the same subscriber number. Though, the network operator has to be sure that the number sent from the gateway is not ported, otherwise the call will terminate in the wrong local exchange (or transit exchange for long distance calls). The gateway must be able to handle these issues.

- The integrity is harmed if the network runs the risk of ending up in a state of livelock or deadlock leading to that calls or requests are halted.

Type of Subscriber Access

There are some integrity issues associated with different forms of subscriber access line to the telephone network. One such difference affecting application services on a Parlay gateway in the Skanova network is the configuration of *Delayed Release of Called Party* (DRCP). The DRCP means that the call is not torn down directly when a callee hangs up if the caller still is off hook. The callee has the possibility to lift the hook off again during one minute if he wishes to continue the call. This is useful if the callee answers in one telephone at home and e.g. wants to change to a telephone in another room. PSTN subscriber access lines have DRCP but ISDN access lines do not.

The DRCP in conjunction with INAP *Capability Set 1* (CS1) affects the possibilities to do application service controlled call forwarding, since CS1 cannot tear down call legs in an established call. So if a call from a caller to a original

[6]From a network point of view the gateway appears as a SCP, see chapter 5.
[7]The NAT is considered as an SCP.

callee is to be redirected to a second callee[8], the call leg to the original callee has to be released by the original callee hooking up. If the original callee then has a PSTN access line the call leg will not be released until after about one minute and first after that it is possible to create a new call leg to the second callee. Consequently, the call would be forwarded but with this unacceptable delay.

This could be an integrity problem, since this somewhat unexpected behaviour might not be foreseen by a service application, and therefore cause some other undefined attendant phenomena. The problem is also stressed by the fact that it is not determinable in advance which access type (PSTN or ISDN) a subscriber number has and hence its characteristics are unknown.

A solution to the problem would be to require ISDN accesses since they do not have DRCP, as stated above. Another way to work it out is to set the parameter `tSuspendTimer` [ETSI, 1994] in the `connect()` operation in INAP. The `tSuspendTimer` determines the delay time until the local exchange terminates the call after the callee hangs up. In the Skanova network this is 100 seconds as default. To enable call forwarding it must be set to maximum a few seconds. The `tSuspendTimer` parameter is not included in the Parlay standard and thus, cannot be set from an application on an application server.

A third solution would also be to upgrade the network to CS2 since it includes tools for so-called *follow-on-calls*. Then it would be possible to tear down a call leg even if the callee (or caller) has not hooked up.

- The integrity is harmed if service applications do not get the expected functionality from the network, and tries to issue instructions that cannot take effekt and thus lead to undefined behaviour.

Restrictions in Underlying Protocol

The gateway receives calls of methods via the Parlay API and translates these calls to appropriate signals, e.g. to SS7 protocols such as INAP. Depending on which signalling protocol and which version is used, not all Parlay functions map to the underlying signalling protocol. This is the case for the CS1 version of INAP that Skanova uses in its network. Thus multiparty calls are not possible to set up with CS1. The gateway must make sure that Parlay functions, that do not map to the currently used signalling protocol, cannot be issued by the service applications. As we saw in the *Type of Access* description on page 49, follow-on-call functionality is restricted by the underlying protocol INAP CS1.

- If application services do not know the limitations in the network and tries to issue impossible instructions to the gateway, the integrity is harmed since the instructions will not take effekt.

7.2.2 Charging

If the network ends up in a state where no means for charging are available, the network owner will get no payment and the integrity harmed. This issue is a very complex one. Skanova, in capacity of a network wholesaler, must always be able to charge other operators for their traffic generated in the Skanova network.

[8] An example of service that would use this kind of forwarding could be a directory enquiries service with call forwarding.

Therefore it is not possible to hand over the charging to third party applications connected via the Parlay gateway. Data for all call leg set-up must be generated by Skanova. Thus, all mechanisms for charging supplied by Parlay for applications can not be used. However, it would be useful if fees not related to call set-up, i.e. fees for services could be put on the ordinary telephone bill, especially for micro payments. This will probably be possible in the future, but Skanova is not there yet.

Charging is not an easy task, since there is a considerable number of systems that has to co-operate. The *Universal Mobile Telecommunications System* (UMTS) operator Hi3G is building a complex system with supercomputers to be able to charge for all the different services planned [Alpman, 2002]. If the Parlay gateway concept is successful, Skanova must probably do the same, then the billing system of today must be replaced or modified to be able to handle payments for services in an easy and uniform way.

An alternative to the micro payment model is if the service providers themselves were charged for calls they generate, i.e. by Skanova. Then the service providers could charge the subscribers of their services directly. This would solve a lot of problems for the network operators, but the customers would get several bills from different operators and service providers.

Lack of Traceability and Surveillance

One important aspect of charging is the ability to detect and trace faulty calls. When such a call is detected it should be possible to change the configuration of the gateway to prohibit further abuse. To be able to track and trace events, all events should be logged. One way to store all call events could be to make *Toll Tickets* (TT) for all call legs, no matter if someone has to pay for them or not. Another advantage of this is that it is possible to decide what call legs to charge for, afterwards.

- It is important to be able to keep track of events in the network, otherwise there will be impossible to know if illegitimate events take place in the network.

Restrictions for Foreign Numbers

In the gateway it might often be necessary to be able to see from which network a call is originating. For example the Directory Enquiries Service calls cannot originate from an arbitrary network if the call is to be forwarded to an enquired number. This is illustrated in figure 7.2 on page 52. The calling party must reside in the Skanova network since it will be hard to charge correctly if the call is originating from another network[9], see section 2.4.

The problem is that due to number portation between operators it is not possible to determine the network of origin by analysing the number of the caller. Unfortunately, there is no other information easily available either to determine the network of origin. This information must be provided in some other way, e.g. from a number database that contains information of which numbers that belong to which network operator.

[9]Note that calls not originating in the Skanova network can still enquire numbers via the service, but the call cannot be forwarded since it is not possible to charge the call.

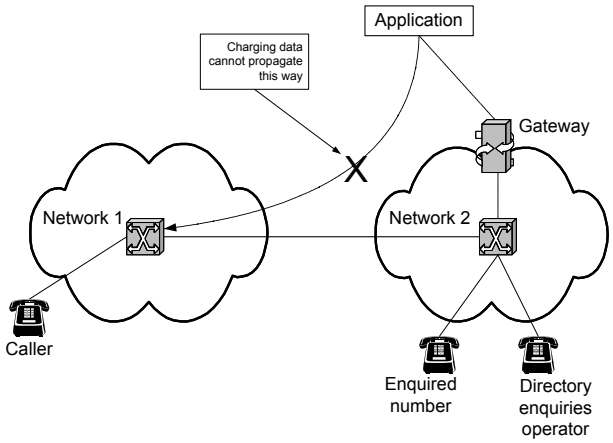


Figure 7.2: Call forwarding restrictions in the Directory Enquiries Service

However, the best solution would be if all other networks had a Parlay gateway as well. Then the directory enquiries service application could connect to those networks and issue charging data directly in the network of the caller.

- The integrity is jeopardised if charging data cannot be transferred between different network operators and thus calls cannot be charged.

Charging Records

In many cases the gateway must make sure that call data records, TT, or other forms of charging data are generated. Otherwise many calls would not be charged properly, or worse, not charged at all.

Another reason for generating charging data is that Skanova, or any network operator, must have basic charging data to be able to charge other operators with whom they are exchanging traffic.

An example is the *Virtual Call Centre* (VCC) service. The basic principle for this kind of service is that the caller should be charged for the call based upon the number dialled and his location. The call is then forwarded to an VCC operator in some network. If the call ends up in a network with higher rate (e.g. VCC telephone operator is overseas or resides in a mobile network) the VCC service provider should be charged for this. It is not defensible to charge the caller for this additional cost, since he has no influence over where the VCC service provider places their telephone operators. The problem is that, by default, no TTs are generated by the network for this excess part of the call. Thus, the basic charging data must generated by the gateway. To avoid possible integrity problems, the gateway operator might want to be able to prohibit the VCC telephone operators from using certain access numbers or networks.

For application initiated calls, charging has to be taken with extra care. As mentioned in chapter 2, TTs are only generated for calls originating in the net-

work, and on the outgoing call legs in the local exchange. Since application initiated calls do not originate in the network the traditional way, no charging records are generated for this kind of call.

- The integrity is harmed if no charging data is generated, and thus calls cannot be charged.

Restrictions for Application Initiated Calls

Restrictions regarding "originating" or "terminating" network might apply to application initiated calls. Depending on the charging model for the "click-to-dial" service (see section 7.1.3) the call might not be allowed to originate or terminate in an arbitrary network. The crucial point is that the party to charge must reside in the Skanova network to facilitate charging. Depending on the charging model this can be the caller or callee. If the Application Service Provider is charged for all application-initiated calls instead, the call can of course originate or terminate anywhere since no subscriber charging has to be done in the network.

- The integrity is jeopardised if it is not defined who to charge.

7.3 Summary

The examples of issues above, that could possible jeopardise the integrity of the telephone network, are of various kinds. Different considerations must be done for the different cases and different technical solutions are required. In chapter 8 we will give our recommendations of which techniques to use, to be able to handle a wide range of issues in a flexible manner.

In table 7.1 below, we can see which integrity issues we have found our example services to have. Some issues must, as seen, be solved per service application, other are general for all possible service applications. One conclusion we can make, based on the integrity issues we have found, is that services in one service category do not have more in common than they do with services in other service categories. Thus it is important to investigate each service by itself.

	Gateway access	Feature interaction	Number analysis	Notifications	Livelock	Charging data	Type of access	Underlying protocol	Network of origin
Virtual call centre	X			X		X	X	X	
Directory enquiries	X			X	X	X	X	X	X
Supplementary service	X	X	X	X	X	X	X	X	
Blacklist	X	X	X	X				X	
Click-to-dial	X			X		X		X	X

Table 7.1: Integrity issues per application

Chapter 8

Managing Integrity Issues

The gateway is to be placed in the Skanova network environment, which by tradition has very high integrity. One explanation to the high integrity is its isolation from other systems and the fact that it has been operated by one monopoly authority with total control of all its components[1]. The gateway opens up access to the signalling network to third parties. This makes the matter of integrity sensitive and the demand to be able to maintain and manage integrity even higher than for traditional network components.

In this chapter we will analyse what it means to have integrity, and what techniques are available to preserve it. We will look closer into techniques that we have found relevant, and then give some examples of how they can be realised in a gateway.

8.1 Integrity Enforcement Model

This section will describe the method to solve the integrity issues. The maintenance of integrity is really about having a set of adequate rules to follow. If these rules were complete[2] and followed, all integrity problems would be solved. One problem is to find all rules before a system is launched, thus new rules might be necessary when problems are discovered. A rule is often of the kind: "if condition, then action". The enforcement of these rules can be divided into two parts:

- *Preconditions:* To know when a rule is to be applied one must know when the preconditions are fulfilled, i.e. one must see that sufficient information about the preconditions must be obtainable, both for known rules and for future ones. For example, since it is sometimes necessary to base decisions on the dialled number, the dialled number must be available.

- *Actions:* One must also be able to execute and enforce a desired action, e.g., prohibit a certain application to connect calls to numbers outside a predefined range. The more actions that are available, the more powerful and flexible rules are possible to build.

[1] Skanova is the successor to Televerket and Telia regarding the responsibility for the main part of the Swedish fixed telecommunication network.

[2] The word *complete* refers to that all rules possibly needed are found.

This theory can be transformed into practice. There are some different techniques and approaches on how to realise the rules and actions. Different vendors[3] of Parlay gateways have chosen different approaches, and combinations of techniques. In the sections 8.2 and 8.3 below, we examine these techniques more closely. Some companies seem to use the Parlay *Service Level Agreement* (SLA) tools and then add some necessary logic to handle issues not solvable with Parlay. Another approach is to use a technique called *Policy Management*. We will examine both these techniques since they are available and most common on the market. Other solutions exist, but since they are proprietary to one manufacturer and not a common technology they are of less importance to study.

8.2 Tools in Parlay

The designers of Parlay have thought of the integrity problems that can arise in connection with the opening of the signalling network. Therefore Parlay has a number of built in mechanisms that contribute to maintaining integrity. Parlay has mechanisms to authenticate applications that connect to the gateway. This ensures that no unauthorised applications can access the signalling network via the gateway. The signalling network itself has no access control since it is designed to be a proprietary network. For example, if applications were directly connected to the *Signalling System No 7* (SS7), it would be harder to control their access. The Parlay API specifies how to restrict the applications access to certain *Service Capability Features* (SCF) in the network, see chapter 4. According to the specification it is possible to decide which SCFs the applications are allowed to access. Within each SCF, there is a number of so called *Service Properties* that can be set to restrict certain possibilities or functionalities, per application. For example, one application might have access to only two of five available SCFs, and in the SCF for Multiparty Call Control the application only has the possibility to set up calls with a maximum of five legs (for example, of sixteen possible). These various tools in Parlay are described in detail below.

8.2.1 Service Level Agreements

A *Service Level Agreement* (SLA) is a contractual agreement between a gateway operator and an application provider. Some of the terms in the agreement are of a technical nature and may restrict the application's rights to the gateway or its resources. Examples of SLA parameters are:

- Not more than one hour planned lack of service per year
- Available SCFs
- Set of methods available for each SCF

[3]Examples of companies retailing Parlay gateways are: Incomit, Ericsson, and Alcatel.

- Format of data and information
- Restrictions on calls or requests, etc.
- Overall capacity and throughput
- Authentication method and key length

These terms have to be enforced in the gateway somehow. Some of the possibilities, to set restrictions, are defined in the Parlay specification and some are not. Some restrictions are just implied, and left to the developer of the gateway to implement. Some of the terms in a SLA can also be there to avoid integrity problems. Fore example, if some calls are not possible to charge for, the SLA will state that these types of calls are prohibited.

The SLA is also a legal document, which can be used to punish the *Application Service Provider* (ASP) if the terms of the SLA are broken. I.e. even if it is not possible for the network operator to look after that all the SLA parameters are followed, the SLA can be used to punish ASPs violating the terms of the SLA. It could be satisfactory to regulate less harmful integrity problems this way, i.e. not implement the possibility to prevent the unwanted behaviour in the gateway. However, if the terms of the SLA are broken they may issue a fine, terminate the contract, or take the ASP to court.

Online SLA refers to the concept of an enterprise operator [Parlay API, 2001] who is allowed to arrange SLAs for the applications in its domain. This concept seems not to be used by any network operators, since it gives control of the access to the services (SCFs) of the network operator to a third party enterprise operator[4]. We will therefore only look into the so-called off-line SLAs. From this point and on we will mean off-line SLAs, whenever referring to SLAs.

8.2.2 Important Interface Classes

To understand how the SLA is enforced in the gateway it is necessary to look at the different classes within Parlay. This will also enlighten us as to the possibilities that Parlay offers, for example, to handle integrity issues. In the following sections we will look at these classes. In section 8.2.3 we will then see how these classes can be used to set up a connection and establish the SLA. These classes, and all other Parlay classes, can be found in the Parlay specification [Parlay API, 2001], as well as all the methods, data types, exceptions, etc. Figure 4.1 illustrates how the different interfaces are related.

[4]A third party enterprise operator can manage SLAs for applications in its domain.

Framework to Application Interface Classes

These classes are selected because they are provided by the framework, and they are used by applications and services, e.g. for authentication and service discovery.

`IpServiceDiscovery class`
The service discovery interface consists of four methods. Before a service can be discovered, the application must know what services the framework supports. The `listServiceType()` method returns all service types currently supported. The `describeServiceType()` returns a description of each service type. An application can discover a specific set of registered services, that both belong to a given type, and possess the desired property values, by using the `discoverService()` method. The framework must only return services that the application is allowed to use. Some applications might not be allowed to use the full functionality (i.e., limitations in property values can exist) and if the application has requested a value above it's limit, the service should not be returned in this case either. The gateway must support this functionality and the limitations must be possible to enter via the gateway's management system. If applications try to use an SCF it is not allowed to, then the `P_ACCESS_DENIED` exception should be raised.

`IpAppServiceAgreementManagement class`
The following methods are invoked by the Parlay framework, and when the `initiateSignServiceAgreement()` method is received from the application, the framework uses `signServiceAgreement()` to request the application to sign the agreement for the service (SCF). The framework provides the agreement text to be signed by the application. The `agreementText` can be set, e.g. to the SLA text itself. The Ericsson Jambala [Jambala, 2002] Parlay gateway and Ericsson Parlay simulator [Parlay Simulator, 2002] uses the string "IagreeToUseTheService", which is a string without meaning and used only because the Parlay standard says so. The Incomit Movade network service platform[5] sends the actual SLA.

When signed, the application can get a reference to the service. If an ASP subscribes to services, the SLAs are handled in other ways, i.e. on-line SLAs, which are not considered in this report. RSA[6] encryption is used, i.e., with one public, and one private key. Several possible key-lengths are possible. In practice the gateway will be protected by firewalls and placed in a secure environment. The applications will connect to the gateway, through the firewalls, via *Virtual Private Networks* (VPN). By using RSA encryption, the Parlay framework can be sure which application is accessing the service. The authentication procedure's main goal is the `signServiceAgreement()`. The agreement text itself is usually not important, it is just a part of the authentication process. The whole process of signing a SLA is of a legal nature. Since the application provider has digitally signed the agreement he is held responsible for the way the application uses the service. A digital signature can be as valid as a signature on plain paper.

[5] A Parlay gateway, see http://www.incomit.se/

[6] RSA is a public-key encryption system based on the factoring problem. Please see http://www.rsasecurity.com/rsalabs/

`IpServiceAgreementManagement class`
The following methods are invoked by the application. To select a service the application can send the `selectService()` method to the framework. It returns a service token identifying the service. Then, in order to access the service, a SLA must be signed. To initiate this process the application can invoke the `initiateSignServiceAgreement()` in the framework (see IpAppServiceAgreementManager below, on page 58).

Framework to Service Interface Classes

These classes are provided by the framework and are to be used by services e.g. for registration of interfaces.

`IpFwServiceRegistration class`
Before a service can be brokered (discovered, accessed, etc.) by an application, it has to be registered with the framework. The process of making a SCF available consists of two steps. First `registerService()` and then `announceServiceAvailability()`. The first step is for registering the service, i.e. to informing the Parlay framework of its name and its specific property values. The framework uses this information when an application calls the method `IpServiceDiscovery()` to find a service with specific property values. The `registerService()` method described above, does not make the service discoverable. The `announceServiceAvailability()` method is invoked after the service is authenticated and its *Service Instance Lifecycle Manager* is instantiated at a particular interface. This method informs the framework of the availability of the Service Instance Lifecycle Manager of the previously registered service. After the receipt of this method, the framework makes the corresponding service discoverable. Per service instance, there also exists a *Service Manager*. Each service implements the `IpServiceInstanceLifecycleManager` interface. This interface supports a method called `createServiceManager()`. When the SLA is signed by `signServiceAgreement()`, the framework calls the `createServiceManager()` to create a Service Manager and returns this to the application.

`IpServiceInstanceLifecycleManager class`
The `IpServiceLifecycleManger`[7] is used to allow the Parlay framework to get access to a Service Manager interface of a service. During the `signServiceAgreement` it is used to return a Service Manager interface (e.g.`IpCallControlManager` for the call control SCS) to the application. Each service has a Service Manager interface that is the initial point of contact for the service. The framework invokes the `createServiceManger()` method to return a Service Manager for the specified application. The service instance will be configured for the specific application using property values agreed upon, in the SLA. These properties and the identity of the application are sent as arguments in the `createServiceManager()` method.

[7]The `IpServiceLifecycleManger` is also called *service factory* in for example the Ericsson report [Moerdijk and Klostermann, 2001].

`IpFwServiceDiscovery class`
This class is not relevant to the SLA establishment, since all services (SCF) are returned, even the ones that the application, requesting them, is not allowed to use. The Parlay specification seems to be a bit unclear about this, and different papers on the subject have come to different conclusions. The Incomit Movade Parlay gateway works this way, i.e. it returns all services.

8.2.3 Connecting new Applications

In table 8.1 we can see step by step how a connection between an application and the Parlay framework in the gateway is established. Then how the restrictions in the SLA are established. This description will illustrate how new service capabilities (SCF) are added, how new applications can get access to the gateway, how applications are granted permission to the service cabability features (SCF) of the gateway, and how the usage of the SCFs may be restricted.

Register the SCS

1. Authentication of SCS
2. Request registration interface from framework
3. Register Lifecycle Manager

Finding services (find SCFs on an SCS via framework)

4. Authentication of application
5. Request discovery interface from framework
6. Discover services in framework

Selecting and accessing a chosen service (SCF)

7. Select service in framework and sign SLA with framework
8. Create Service Manager in the Life Cycle Manager
9. Return Service Manager to framework
10. Return Service Manager to application

Using the service

11. Use service (use SCF)

Table 8.1: Connecting new applications to the gateway

We will now examine the steps in table 8.1 in detail. First the SCS will contact the framework and request the registration interface (1, 2). It will then use this interface to publish its capabilities (*Service Property Values*) and a reference to its *Service Instance Lifecycle Manager Interface*, or shorter, Lifecycle Manager (3). The Lifecycle Manager allows the framework to request the SCS to create an SCF instance. The framework and the SCS know each other from here on.

When an application wants to use a service, the application contacts the framework, gets authenticated (4), and requests the discovery interface (5). The framework returns a reference to the discovery interface and then the application uses this interface to request the services it is interested in (6). This can be a three-step process, see section 4.3.2 on page 22. The framework returns references to all the SCFs that fulfil the request of the application.

The application selects one service (SCF) and signs the SLA (`signServiceAgreement()`) (7). The framework then contacts the Lifecycle Manager in the SCS and forwards the conditions (stated in the SLA) under which the application is allowed to use the SCF (8). The Lifecycle Manager then creates a SCF instance that will be used by this specific application. This specific SCF has the properties as they are defined in the SLA. The reference to this SCF instance is returned to the framework (9). The framework then returns the reference to the application (10).

From this moment the application can use the service and all its features. If the service is Generic Call Control, the application for example can create a call with the `createCall()` method.

8.2.4 Service Properties

Each SCF is of a certain service type, which has a number of pre-specified service properties. These properties, can be used to preserve the integrity, and prohibit different events. The properties can be given specific values at the time of the registration of the SCF. Before an SCF can register with the framework, the framework must implement (i.e. support) the type of service that the SCF is, i.e. it must know what service properties to expect. At the time for establishment of a SLA in the gateway, the registered properties can be restricted and set to the values specified in the SLA. The restriction could also apply to only a specific application or a group of applications and allow them to use only a subset of the property values. These application specific restrictions are set during the creation of the Service Manager mentioned in the *Service Instance Lifecycle Manager Interface* on page 59.

General Service Properties

The following table lists properties that are used in the context of the Service Level Agreement, e.g. to restrict the applications access of the capabilities in the SCS.

Each service instance (SCF) has the following general properties, no matter what its service type:

- **Service Name**
  This is the name of the service, e.g. UserStatus
- **Service Version**
  The version of the API, e.g. 3.0 Service
- **Instance ID**
  This property uniquely identifies a specific instance of the service. Generated by the framework

- **Service Instance Description**
  Textual description of the service

- **Product Name**
  Name of the product, e.g. "Userstatus.com"

- **Product Version**
  Version of the product that provides the service

- **Supported Interfaces**
  List of strings with interfaces names supported by the services, e.g. "`IpUserStatus`"

Specific Service Properties for the Generic Call Control API

The following service properties are specific to the Generic Call Control API. Per application, they can be set to different values to restrict applications' access to the Generic Call Control functions, e.g. to restrict which numbers the application is allowed to trigger. These properties can be used to enforce integrity regarding call set-up, other APIs have other properties that can be useful to solve other issues. This report will be limited to only investigating issues related to call set-up, see section 1.4. Each property has a name and a type. The type can often have a set of possible values. In the descriptions below we will see what set of values different properties have:

- **Triggering addresses**
  `P_TRIGGERING_ADDRESSES` indicates for which numbers notifications[8] may be set. For termination notifications it applies to the terminating number, for outgoing notifications it applies to originating number.

  This property sets an address range (or ranges) on which the application can trigger. A SLA can restrict this range. By invoking a trigger, the application can gain control of calls to or from a certain subscriber number via `enableCallNotification()`.

  `P_TRIGGERING_ADDRESSES` is of the type `ADDRESS_RANGE_SET`. For the Skanova PSTN network it always concists of `E.164`[9] numbers, e.g. for a personal number series the address range could be all addresses matching "4675600????", i.e., 10000 numbers, where "?" stands for one digit. "*" can also be used as a joker sign and stands for an arbitrary number of digits.

[8] A notification is when an application is informed about some event in the network by the gateway, e.g. off-hook on a specific subscriber number.

[9] ITU-T Recommendation E.164. The international public telecommunications numbering plan, e.g. "4687131000" is an E.164 number. The `E.164` address format is an international number without the international access code, including the country code and excluding the leading zero of the area code.

- **Notification types**
  P_NOTIFICATION_TYPES indicates whether or not the application is allowed to set originating and/or termination triggers. The set of values is:

  - P_ORIGINATING
  - P_TERMINATING

  This property is used to define if it is possible to trigger on originating or terminating numbers. In the Skanova network it is only possible to set triggers on originating calls since the call terminates in a local exchange without *Service Switching Point* (SSP) functionality. The terms *Originating* and *Terminating* can be a confusing. A termination trigger can be an *answer* event on a specific subscriber access number. However, it is still possible to trigger on termination events in the Skanova network, e.g. answer, but then the application must be in charge of the call. If a call is triggered with an originating trigger, the call will be treated as an *Intelligent Network* (IN) call.

  In other networks, such as mobile networks it is possible to trigger on terminating events, e.g. two mobile telephones with the same number (twin SIM-card) can ring simultaneously until someone answers in one of the telephones. Then the trigger is activated and informs some service platform, e.g. of which telephone answered, etc.

- **Monitoring mode**
  P_MONITORING_MODE indicates whether or not the application is allowed to monitor in interrupt and/or notify mode. The set of values is:

  - P_INTERRUPT
  - P_NOTIFY

  In interrupt mode, the SSP notifies the gateway and then waits for an answer from the gateway by means of e.g. a telephone number to redirect the call to. If in notify mode the SSP will only notify the gateway of events in the network and will not wait for instructions from the gateway to complete the call.

- **Numbers to be changed**
  P_NUMBERS_TO_BE_CHANGED indicates which numbers the application is allowed to change or add for legs of an incoming call. Allowed sets of values are:

  - P_ORIGINAL_CALLED_PARTY_NUMBER
  - P_REDIRECTING_NUMBER
  - P_TARGET_NUMBER
  - P_CALLING_PARTY_NUMBER

  If an application is allowed to change the calling party number, it could for example falsify who is calling and affect charging, etc. However, in some cases this could be a useful feature, e.g. if an company want the switch-

board number to be shown to the caller, and not the specific extension. Applications doing number translation must be able to change the original called party number to redirect the call to a new destination.

- **Charge plan allowed**
  `P_CHARGEPLAN_ALLOWED` indicates which charging model is allowed in the `setCallChargePlan` indicator. To avoid integrity issues concerning charging, the gateway must facilitate charging. Generally, the control over basic call charging cannot be handed over to a third party, since they are not trusted. It would also expand the integrity problem to hand out the call charging process to a third party. However, supplementary charging features could use this solution. The allowed value set:
  - `P_CHARGE_PER_TIME`
  - `P_TRANSPARANT_CHARGING`
  - `P_CHARGE_PLAN`
- **Charge plan mapping**
  `P_CHARGEPLAN_MAPPING` indicates the mapping of charge plans to a logical network charge plan indicator. The use of charge plans via Parlay enlarges the integrity problems and is therefore, not in scope of the report.

8.2.5 Modified Service Properties

One way to make the Parlay Service Properties more powerful, is to add new ones. With this approach it might be possible to solve some more problems, but still the method has a number of disadvantages.

The available set of Service Properties must be pre-programmed and cannot be modified afterwards, i.e. if new a new problem appears and a new service property is needed, the gateway software has to be modified and recompiled. As with the built in Service Properties it is only possible to prohibit events, no actions can be taken. Finally, this solution extends the Parlay API and makes is proprietary, i.e. an application made for this API is not might not be portable to other gateways and this is somewhat against the idea of having a standard gateway.

8.3 Policy Management

Policy management is a technique based on rules such as “if condition, then action”. It is a flexible system, and policy rules can be added, modified, or removed on the fly. Thus, it is not necessary to recompile the gateway software as in the case with adding Service Properties.

This section will analyse what opportunities Policy Management provides to protect the network integrity. As we have seen in chapter 4 on page 27, there is something called Parlay policies. Parlay policies is an API that opens up some policy functions via an API for applications to use. However, since this does not strengthen the integrity, quite the contrary, we will not look into the Parlay policy API. Open policy APIs are referred to as on-line policies. Policies for internal management in the gateway are refereed to as off-line policies. Whenever taking about policies from this point and on, we refer to off-line policies.

8.3.1 Background

The policy information model is based on the IETF Policy Core Information Model [Moore et al., 2001] and Requirements for a Policy Management System [Mahon et al., 1999]. Policy management is becoming more and more commonly used for the purpose of managing complex systems. Companies such as Ilog with its product "ILOG Jrules" have highly developed tools to create rules, and engines to make the policy decisions. The policy management model is used in a variety of contexts, not only in tele and data communication. Or as Ilog puts it [ILOG Jrules, 2002]:

> "Rules for business applications can be found in any business domain that enforces dynamic and frequently changing statements of business policy in application code. Today, many industries are benefiting from business rule technologies, enabling companies to quickly respond to markets and customers."

8.3.2 Policies

A policy is a rule that defines behaviour of a system derived from SLAs and enterprise policies. A policy can change the behaviour of a system without modifying the implementation. Policies may be dynamically modified. A policy consists of one or more conditions, and one or more actions, see figure 8.1.

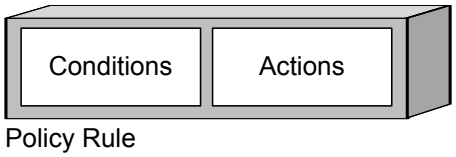


Figure 8.1: A policy rule

Policies are set by the network operator, and must be preformed by third party service applications. Policies are often set-up to limit the network access for applications or to perform some action when specified preconditions are fulfilled. Policies can be used to solve integrity issues, e.g. generate charging data or prohibit unwanted call set-up.

A policy can restrict who is allowed to access a service, what operations may be performed when, and what should be done when a specified event occurs.

8.3.3 Architecture

The policy management architecture can be seen in figure 8.2 on page 66. It consists of a number of logical entities that are described below. Where these entities are to be placed and how they interconnect are of less importance. The important thing is that they all reside in the network operator's domain and are under the network operator's control.

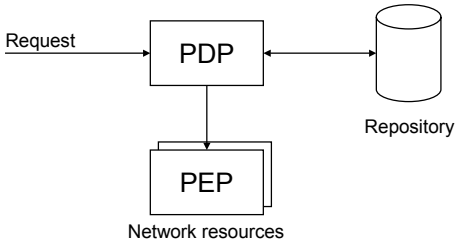


Figure 8.2: Policy management architecture

8.3.4 The Network Policy Engine Complex

The network policy engine complex resides within the network operator's domain. The complex has three logical elements, the *Policy Repository* (PR), the *Policy Decision Point* (PDP), and the *Policy Enforcement Point* (PEP). These three are the foundation of a policy-based network.

Policy Repository

The Policy Repository is where all policy related information is stored. The repository is linked to the Policy Decision Point. Rules can be added or removed from the repository on the fly. Rules can also be grouped together to provide an overall functionality, e.g. for a certain service. The same rules can be applied to one or more services.

Policy Decision Point

The Policy Decision Point is where policy decisions are identified based on the policy rules supplied from the repository. State information from the network may also be used as a basis for decisions. The more information supplied to the PDP, the more powerful rules that are possible to create since more event criteria are available.

Policy Enforcement Point

One Policy Enforcement Point is needed for every network resource. The PEP enforces policies identified by the PDP. The policies are enforced in the particular network resource the PEP is running on behalf of. The network resource could range from a billing system or user account management systems to exchanges and routers.

Network Resources

The network resources are entities to which policies are applied by a PEP. Network resources can range from high level billing systems to core network components, such as routers and exchanges. The network components use a variety of different protocols and communicate with the PEP. One example is if a policy is to create a *Toll Ticket* (TT), then the PEP will enforce this by sending an instruction[10] to an SSP to create a TT.

8.3.5 Example of Policy Engine Driven Execution

How a policy is executed in the policy complex is illustrated in figure 8.3 in conjunction with the enumerated list below.

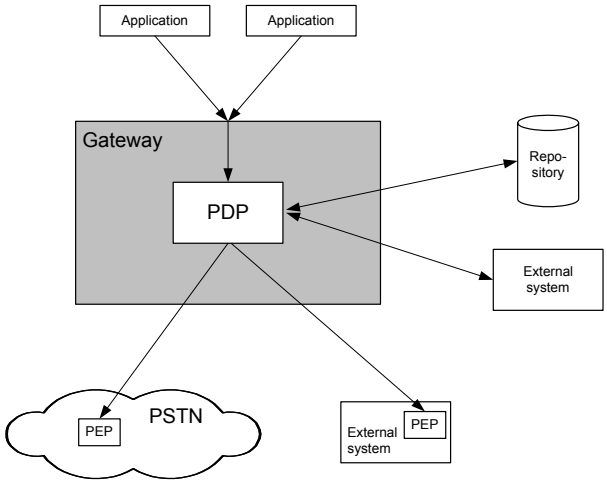


Figure 8.3: Policy engine driven execution

1. A service application sends a call set-up request to the gateway. The request ends up in the PDP.

2. The PDP retrieves policy rules from the policy repository database.

3. The PDP makes a decision based on the retrieved rules and, if necessary, information retrieved from external data systems.

4. A decision is sent to the PEP in the network. The PEP executes the decision, in this case the application is allowed to access the telephone network.

5. Other policies related to the call might be enforced in external system such as billing or logging systems.

[10] Could be the INAP signal: `FurnishChargingInformation()`.

8.3.6 Integrity Issues to Address

In chapter 7 we have given examples of integrity problems that arise in conjunction with different services. One of the services described in section 7.1.1 on page 44 was the Directory Enquiries Service. We will use this service as an example and see how some integrity issues specific for this particular service can be solved with policy management. The integrity issues we will look at are:

- *Charging data:* The gateway must ensure that all calls are charged correctly. When a call first is placed to the Directory Enquiries Service number, an incoming call leg from the caller is set up to an exchange (SSP). The exchange asks the application, via the gateway, for instructions to continue processing the call. A new destination number is given by the application, in this case the number of a telephone operator, and the exchange sets up a new outgoing leg to that number. A TT is automatically generated for the incoming leg in the local exchange of the calling party. For the outgoing leg no TT is generated. If the call is forwarded to the enquired number, a new outgoing leg is set-up. No TTs is generated for this leg either, i.e. for the last two cases the gateway must produce charging data.
- *Originating network:* The possibility to be connected to the enquired number is open for callers with Skanova subscriber access lines only, since it is not possible to charge other callers. Attempts to forward other calls must therefore be stopped.

This gives us three events and thus three rules to create. We conclude with the following points:

1. Create charging data for the call leg to the telephone operator
2. Create charging data for the call leg created due to the forwarded call
3. Prohibit calls from non-Skanova subscriber accesses to be forwarded

8.3.7 Policy Rules

When creating rules for the three events above we must think of both what preconditions are needed and what actions must be taken. When designing the system one must see to that these preconditions are available to the PDP as a basis for its decisions. The same line of reasoning applies to the actions. The needed actions must be executable by the PEP.

Rule 1

For the first case the rule could be:

If
the application is “Directory Enquiries Service”
and call not previously routed
and status is pending

Then
set status to “proceed”
and issue a `furnishChargingInformation` for the new call leg

Rule 2

For the second case the rule could be:

If
the application is "Directory Enquiries Service"
and call previously routed
and call originating in Skanova network
and status is "pending"

Then
set status to "proceed"
and issues `furnishChargingInformation` for the new call leg

Rule 3

For the third case the rule could be:

If
the application is "Directory Enquiries Service"
and call previously routed
and status is "pending"
and call not originating in Skanova network

Then
deny request

8.3.8 Scalability

The gateway need have a number of rules that applies to all applications. These rules are of constant number irrespective of the number of service applications connected. How these rules are formulated depends on how the gateway is implemented, and it is therefore hard to estimate the number of rules needed. However, less than 100 common rules seems reasonable with respect to the gateways we have studied.

All service applications connected to the gateway must also have a set of rules that are specific for each service. An example of this is explained in section 8.3.7 above. Thus, the number of rules will increase with the number of service applications connected. Based on the example above, our estimation is that, in average, a service application will need about ten active rules. Of course this is depending on how complex the service is.

One advantage with policy rules is that they can be reused, i.e. policies for one application can also be used for another. E.g. all Directory Enquiries Services may use the same set of rules. This will lead to a more efficient system, and the number of rules in the repository will not grow when new applications of an already existing type are added. It might even be possible to have a relatively small set of rules that can be combined in groups for a specific service type. This way the total number of rules may be kept low.

At gateway system start-up, when new applications are added, the number of rules will increase quite fast. This is a consequence of that almost every added application, in the beginning, will be of a new type, and therefore will need application specific rules that do not already exist. Later, when several

applications are added, the rules can be reused and the increase of rules per new application will be lower.

If many rules simultaneously are active, and the number of requests is high, the system may become overloaded. It must be ensured that the gateway has the computing power necessary to support the number of active rules. The computing power needed to handle the policy complex is the biggest drawback to Policy Management.

8.4 Pros and Cons

Both Parlay SLAs and Policy Management each have their advantages and disadvantages.

8.4.1 Parlay SLA

We have found the built-in Parlay SLA capabilities to be useful to solve some of the integrity issues, but far from all. For example, as shown, access and authentication problems may be solved and certain applications may be prohibited from using SCFs and methods. The Service Property "triggering addresses" can be used to restrict triggering criteria, and the "numbers to be changed" property can be used to see to that applications doing call-forwarding does this in a way they are allowed to, i.e., the only functionality provided by Parlay SLAs is to prohibit certain events, no actions can be taken. Many of the problems in section 7.2 require that some actions are taken, e.g., create charging data. A more extensive and flexible approach is needed to solve the majority of the integrity problems.

8.4.2 Policy Management

Policy rules can be a very powerful way of managing a system or network. Rules can be designed to prevent events that could harm the network integrity, or force actions to be taken to ensure its integrity. To realise a powerful policy based management, the network operator has to provide the PDP with e.g. information about as many events and states in the system as possible. The PDP could, for example, be connected to the subscription system of the network operator's to base decisions upon the type of subscription the users have.

Since policy rules can be modified and added in runtime to the policy repository, it is very easy for a gateway operator to quickly add or change a rule. This is very useful when security breaches are discovered, since they can be fixed quickly, and without shutting down the gateway. As we could see in the examples of policy rules in section 8.3.7 problems with charging are easily solved. As long as sufficient preconditions are supplied, almost all integrity problems mentioned in chapter 7 can be solved with help of Policy Management.

The risks with using Policy Management is that if rules are added without thought, the number of rules will be large, thus requiring a lot of computing power. This may lead to long response times and poor performance in the gateway. But if rules are added moderately this need not be a problem.

8.5 Integrity Management Conclusions

Based on the pros and cons above we suggest that the authentication mechanisms in Parlay should be used to ensure that only authorised applications connect to the gateway. However, we regard the control provided by the built-in Service Properties in Parlay, to be quite powerless.

Policy Management on the other hand has a number of possibilities to solve the integrity issues mentioned in this report, as long as the gateway is implemented to provide needed preconditions and actions. Neither should there be problems supplying the needed computing power to ensure a smooth performance as long as policy rules are added moderately, and existing rules are reused as much as possible.

Chapter 9

Conclusions

The issue for this thesis has been the integrity problems that arise, and the solutions needed, when introducing a Parlay gateway in a PSTN network. In chapter 6 we have stated our definition of integrity which have been used when determining possible integrity breaches in the Skanova network. In chapter 7 we have investigated and given examples of different integrity issues that can arise, and in chapter 8 we have looked at solutions and techniques to address the encountered integrity issues.

This chapter will conclude and sum up the most important results and recommendations. It will also conclude what further research is needed, and which areas must be investigated closer.

9.1 Conclusions Concerning Network Integrity Definition

We have experienced that our extensive definition of integrity in chapter 6 does not completely match the actual integrity issues we have found. The integrity problems we have concentrated on have been functional or software related problems. Chapter 6 describes several other aspects that have stronger connection to hardware. However, the definition is still valid and useful, and can be used also when evaluating other aspects of the gateway than the ones in this report.

9.2 Conclusions on Integrity Issues

In chapter 7 we studied the integrity aspects of connecting a Parlay gateway to a telephone network. During this thesis we have described a variety of integrity issues. It is hard to categorise, or in other ways see similarities in many of the problems. Among the integrity issues we have found, some issues are of greater importance than other, e.g. problems related to gateway overload. Those problems will jeopardise all services connected via the gateway. However, since we only have seen a small number of potential problems, by studying sample service applications, we do not want to point out the most serious integrity problems, but we can say that most problems arise in conjunction with charging. This is because it is both a question of business, and an issue with a lot of technical aspects. The

problem is often that charging data for a call is lacking or insufficient, if no precaution is taken.

We have come to the conclusion that it is very hard to predict all possible problems that might arise. When new types of services are installed, new problems will surely arise. This rises demands on the technical solution that has to manage the integrity issues, since the solution must be flexible, i.e. able to handle a variety of problems.

9.3 Recommendations for Maintaining Integrity

In chapter 7 we have given examples of different integrity problems. To maintain the integrity of the network, we have to find solutions for each of these and other integrity problems. We have looked for the best technical solution according to our limitations stated in section 1.4 on page 2. This implies that, e.g., economical aspects are not considered. The solutions we have investigated are *Parlay SLA*, and *Policy Management*. We have concluded the following:

- *Parlay SLA* is useful for handling access and authentication issues. There is good support for encryption, and well-defined and standardised ways to gain access to the gateway. It is also important to remember that the SLA itself is a legal document that can regulate what an service application is allowed to do and not. If some clauses are broken, the Application Service Provider may be disconnected from the gateway, punished with an extra fine or even taken to court.

- *Policy Management* may be used to address integrity issues of other kinds than Parlay SLA is able to handle. This could be, e.g., prohibiting certain types of call set-up, creating basic charging data, etc. The advantage with Policy Management is its general architecture and flexible management. It is also very easy to adapt the policy complex to new integrity threats. However, a risk with Policy Management is that the number of rules grows without bounds and the system becomes unmanageable. If rules are added sensitively, and reused as much as possible, this is not a problem.

9.4 Future Work

We have identified areas that need further work. This is work that origin from areas that we have given low priority due to lack of time or because it has been out of the scope of this thesis.

- Inconsistency between the network and the gateway could be a problem, since the gateway and the network could have different ideas of which application that handles a certain call. This is an example of that the call-notification issued by the application fail in its correspondence with the triggers in the network, since today these are not managed via the same system and therefore not mapped. How this should be prevented need further investigation. This includes the study of support systems, etc.

- We have not studied hardware related integrity issues intensively. This is something important, since hardware problems (computational power, compatibility, etc.) most likely could cause severe integrity problems making the gateway non-operational.
- We have found charging to be problematical, since we can not rely upon third parties generation of the basic charging data. However, if it was possible to facilitate charging via the Parlay charge plan interface, a lot of proprietary solutions would be avoided. Service applications might also like to charge customers for genuine services (e.g. information services), not related to call set-up, via the phone bill. Maybe the new Parlay API "Content Based Charging" can be useful for this. These topics need further investigation.
- Loops in the signalling system seem to be handled by the switches in the network, but exactly how this is done, and if this always is the case, need further investigation.
- There are other techniques available for handling integrity issues in the gateway than Parlay SLA and Policy Management. More research is needed in the area.
- Our solutions are based on a technical perspective. To get the complete picture, market analysis and financial estimates must be done. This will probably also reveal that more SLA parameters are needed and it will most likely affect some policy rules. This has to be examined.
- The parlay API as a whole should be studied. There are indications on the market that operators and service providers think the Parlay API is growing too complex. Thus different operators might develop proprietary limited version of the API. This may lead to that applications working perfectly with one gateway, give rise to security breaches on another gateway. The consequences of this evolution should be studied.

References

[Albertsson, 2001] Albertsson, E. (2001). Tjänsterealisering-Nät: Nummerportabilitet fas 2 för Telias fasta nät i Sverige. Technical report, Telia AB. Internal document.

[Alpern and Schneider, 1984] Alpern, B. and Schneider, F. B. (1984). Defining Liveness. Technical report, Department of Computer Science, Cornell University, Ithaca, New York. `http://www.cs.cornell.edu/Info/People/fbs/publications/defining_liveness..eps`.

[Alpman, 2002] Alpman, M. (2002). Hi3Gs superdator ska kolla teleräkningen. *Ny Teknik*, (36).

[ANSI, 2001] ANSI (2001). American National Standards Institute, Telecom Glossary 2000. Web page accessed in July 2002. `http://www.atis.org/tg2k/`.

[Appium, 2002] Appium (2002). Appium Technologies AB. Web page accessed in August 2002. `http://www.appium.com/`.

[Beddus et al., 2000] Beddus, S., Bruce, G., and Davis, S. (2000). Opening Up Network with JAIN Parlay. *IEEE Communications Magazine*, pages 136–143.

[Boman et al., 1992] Boman, R., Heidermark, A., Remmereit, A., and Ritter, C. (1992). Signalering. *Nät- och Funktionsutvecklingsplan för telefonnätet 1993-1997*, pages 23–26. Televerket Division Nättjänster.

[Calder et al., 2002] Calder, M., Kolberg, M., Maghill, E. H., and Reiff-Marganiec, S. (2002). Feature Interaction: A Critical Review and Considered Forecast. Technical report, Department of Computing Science, University of Glasgow and Department of Computing Science and Mathematics, University of Stirling, United Kingdom. `http://www.dcs.gla.ac.uk/~muffy/papers/calder-kolberg-magill-reiff.pdf`.

[Corba, 2002] Corba (2002). Common Object Request Broker Architecture. Web page accessed in August 2002. `http://www.corba.org/`.

[CPP, 2002] CPP (2002). C++ programming language. Web page accessed in April 2002. `http://std.dkuug.dk/jtc1/sc22/wg21/`.

[ETSI, 1994] ETSI (1994). Intelligent Network (IN), Intelligent Network Capability Set 1 (CS1), Core Intelligent Network Application Protocol (INAP). ETS 300 374-1 standard.

[Handley et al., 1999] Handley, M., Schulzrinne, H., Schooler, E., and Rosenberg, J. (1999). SIP: Session Initiation Protocol. Technical report, The Internet Engineering Task Force, IETF. RFC 2543.

[Hansson and Teglöf, 1985] Hansson, L. and Teglöf, B. (1985). *Telekommunikation Telefonnät 1.* Studentlitteratur. Lund.

[Henström et al., 1992] Henström, P., Knutsen-Öy, L., and Valas, D. (1992). Taxering och avräkning. *Nät- och Funktionsutvecklingsplan för telefonnätet 1993-1997*, pages 20–22. Televerket Division Nättjänster.

[ILOG Jrules, 2002] ILOG Jrules (2002). ILOG Business Rules: ILOG JRules: A Rule Engine for Java. Web page accessed in July 2002. `http://www.ilog.com/`.

[IN, 2002] IN (2002). Intelligent Network. Web page accessed in April 2002. `http://www.iec.org/cgi-bin/acrobat.pl?filecode=66`.

[ISUP, 2002] ISUP (2002). ISUP call set-up. Web page accessed in April 2002. `http://www.sunrisetelecom.com/technotes/TEC-GEN-3B-SS7.pdf`.

[Jambala, 2002] Jambala (2002). Ericsson - JAMBALA Service Capability Server (J-SCS). Web page accessed in September 2002. `http://www.ericsson.com/products/product_selector/JSCS_hpprod.shtml`.

[Jarsjö, 1992] Jarsjö, U. (1992). Telefon- och ISDN-näten. *Nät- och Funktionsutvecklingsplan för telefonnätet 1993-1997*, pages 28–32. Televerket Division Nättjänster.

[Java, 2002] Java (2002). Java programming language. Web page accessed in April 2002. `http://java.sun.com/`.

[Mahon et al., 1999] Mahon, H., Bernet, Y., and Herzog, S. (1999). Requirements for a Policy Management System. Technical report, The Internet Engineering Task Force, IETF. Internet Draft.

[Modarressi and Skoog, 1990] Modarressi, A. R. and Skoog, R. A. (1990). Signalling System No. 7: A Tutorial. *IEEE Communications Magazine*, 28(7):19–35.

[Moerdijk and Klostermann, 2001] Moerdijk, A.-J. and Klostermann, L. (2001). Open Service Architecture: concepts and standards. Technical report, Ericsson Eurolab.

[Moore et al., 2001] Moore, B., Ellesson, E., Strassner, J., and Westerinen, A. (2001). Policy Core Information Model – Version 1 Specification. Technical report, The Internet Engineering Task Force, IETF. RFC 3060.

[Orava, 2001] Orava, H. (2001). Nummerportabilitet. Technical report, Telia AB. Internal document.

[Parlay API, 2001] Parlay API (2001). Open Service Access; Application Programming Interface. Final draft ETSI ES 201 915.

[Parlay Group, 2002] Parlay Group (2002). The official web page. Web page accessed in April 2002. http://www.parlay.org/.

[Parlay Simulator, 2002] Parlay Simulator (2002). Ericsson: Ericsson Mobility World. Web page accessed in September 2002. http://www.ericsson.com/mobilityworld/.

[Prnjat and Sacks, 1999] Prnjat, O. and Sacks, L. (1999). Integrity Methodology for Interoperable Environments. *IEEE Communications Magazine*, 37(5):126–139. http://www.ee.ucl.ac.uk/~pants/papers/ieee.pdf.

[Prnjat and Sacks, 2000] Prnjat, O. and Sacks, L. (2000). Interdomain Integrity Management for Programmable Network Interfaces. *Third IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS), Fortaleza, Brazil*. http://www.ee.ucl.ac.uk/~pants/papers/Prnj00b.pdf.

[RMI, 2002] RMI (2002). Remote Method Invocation. Web page accessed in August 2002. http://java.sun.com/j2se/1.3/docs/guide/rmi/spec/rmiTOC.html.

[Rydqvist, 1987] Rydqvist, P. (1987). *Lär känna AXE*. Telefonaktiebolaget LM Ericsson. Stockholm.

[SEI, 2000] SEI (2000). Software Engineering Institute, Glossary. Web page accessed in July 2002. http://www.sei.cmu.edu/str/indexes/glossary/robustness.html.

[Stallings, 2000] Stallings, W. (2000). *Data & Computer Communications*. Prentice Hall International, sixth edition.

[UML, 2002] UML (2002). Unified Modeling Language. Web page accessed in April 2002. http://www.uml.org/.

[Valas, 1999] Valas, D. (1999). TT-record description. Technical report, Telia AB. Internal document.

[Ward, 1995] Ward, K. (1995). The Impact of Network Interconnection on Network Integrity. *British Telecommunications Engineering*, 13:296–303.

Appendix A

Acronyms

Table A.1: Acronyms

ACM	Address Complete Message
ACQ	All Call Query
ANM	Answer Message
ANSI	American National Standards Institute
API	Application Programming Interface
AS	Application Server
ASP	Application Service Provider
ATM	Asynchronous Transfer Mode
BCSM	Basic Call State Model
CDR	Call Data Record
CORBA	Common Object Request Broker Architecture
CPG	Call Progress Message
CPU	Central Processing Unit
CS1	Capability Set 1
CS2	Capability Set 2
CTR	Connect to Resource Connection
DFC	Disconnect Forward Connection
DRCP	Delayed Release of Called Party
ETSI	European Telecommunications Standards Institute
FCFS	First Come First Serve
FCI	Furnish Charging Information
IAM	Initial Address Message
IDP	InitialDP
IN	Intelligent Network
INAP	Intelligent Network Application Protocol
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
JAIN	Java Advanced Intelligent Network
LW-INAP	Light Weight INAP
MTP	Message Transfer Part
NAT	Number Address Translator
NoA	Nature of Address
OSI	Open System Interconnection
OSPF	Open Shortest Path First
P&C	Prompt and Collect User Information
PBX	Private Branch Exchange
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PR	Policy Repository
PSTN	Public Switched Telephone Network
PTS	Post- och Telestyrelsen
PVC	Private Virtual Circuit

QoS	Quality of Service
REL	Release
RLC	Release Complete
RMI	Remote Method Invocation
RSA	Rivest, Shamir, and Adleman
SCCP	Signalling Connection Control Part
SCF	Service Capability Features
SCP	Service Control Point
SCS	Service Capability Server
SEI	Software Engineering Institute
SIM	Security Identification Module
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SLEE	Service Logic Execution Environment
SMS	Short Message Service
SP	Signal End Point
SS7	Signalling System No 7
SSP	Service Switching Point
STP	Signalling Transfer Point
TCAP	Transaction Capabilities Application Part
TT	Toll Ticket
UML	Unified Modelling Language
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resource Locator
VC	Virtual Circuit
VCC	Virtual Call Centre
VPN	Virtual Private Network
WLAN	Wireless Local Area Network