Design of a Teleworking Service Using Parlay Framework Federation

Morayo Akin-Ojo

A research report submitted to the Faculty of Engineering and Built Environment, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, September 2005

Declaration

I declare that this research report is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other university.

Signed this ___ day of _____ 20___

Morayo Akin-Ojo.

Abstract

A teleworking service allows people to work effectively together from home or other approved locations away from the regular work site, on an established work schedule. This is made possible via the use of Information and Communications Technology (ICT). Presently, there are isolated applications that can assist teleworkers, such as e-mail and video conferencing, which were developed for use over the Internet. But the Internet is a best-effort network with no guarantee of Quality of service (QoS), low security and no standard billing system. The design of this teleworking service involves the integration of many existing services like e-mail, messaging, video conferencing, shared whiteboard and database access. Other requirements are for service providers interworking for service and resource usage, security, and QoS specification. Hence, we explore the emerging open service concept to create this integrated teleworking service that can be made available for subscription by corporate bodies and individuals.

Service federation is the interaction between teleworkers across service provider domains. It is achieved via the interworking of providers' services, and is an essential aspect of teleworking. We have realised a service federation in a secure and seamless manner in the OSA / Parlay environment via the use of the OSA / Parlay framework. We looked at the use of a framework federation for the actual implementation of service federation. This framework federation is an interworking of frameworks based on an agreed-upon federation contract between them. New framework interfaces were introduced to facilitate this proposed solution, as the OSA / Parlay specifications do not yet support this approach.

Service composition is the creation of a new service instance by composing one or more other services. We implemented this via the use of framework and trader federation. The trader federation was used to locate services or users in different ASP domains. A high level design of the teleworking service was done with federation explored for actual implementation. The Common Object Request Broker Architecture (CORBA) trading service was used to prove the concept. The RM-ODP methodology is followed in this teleworking service design. The OSA / Parlay terminal capability, generic call control, multiparty and location and Service Capability Features (SCF) were used for implementing in the CORBA Distributed Processing Environment (DPE).

Acknowledgements

First, I would like to thank God for bringing me this far.

The following research was performed under the auspices of the Centre for Telecommunications Access and Services (CeTAS) at the University of the Witwatersrand, Johannesburg, South Africa. This centre is funded by Telkom SA Limited, Siemens Telecommunications and the Department of Trade and Industry's THRIP programme. This financial support was much appreciated.

I would like to extend thanks to my supervisor, Prof. Hu Hanrahan for his guidance and assistance throughout the duration of this research project. The help of Dr. Ian Kennedy was much appreciated. In addition, I would like to thank my colleagues at CeTAS for their valuable inputs during the research. Finally, I would like to thank my parents, siblings and fiance for their love, support and patience, which have helped me throughout my studies.

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
ASP	Application Service Provider
CompUSM	Composition User Service Session Manager
CORBA	Common Object Request Broker Architecture
DPE	Distributed processing Environment
GUI	Graphical User Interface
HLR	Home Location Register
ICT	Information and Communication Technology
IDL	Interface Definition Language
ICT	Information and Communications Technology
IP	Internet Protocol
MSC	Mobile Switching Center
NGN	Next Generation Network
OAM	Operations, Administration and Maintenance
ORB	Object Request Broker
OMG	Object Management Group
OSA	Open Services Access
PAM	Presence and Availability Management
PSDN	Public Switched Data Network
PSTN	Public Switched Telephone Network
QoS	Quality Of Service
RM-ODP	Reference Model for Open Distributed Processing
SAG	Service Assignment Group
SATINA	South African TINA
SCF	Service Capability Feature
SCS	Service Capability Server
TINA	Telecommunications Networking Information Architecture
VLR	Visitor Location Register
VoIP	Voice Over IP

Chapter 1

Introduction

Teleworking is a service that allows people to work effectively in places other than the conventional workplace. The teleworking service has many benefits, which make it desirable service, especially for people who want to work from home.

The Next Generation Network (NGN) is an evolution towards convergence of fixed, mobile and packet based networks. The NGN converges voice, data and video services into one network. An objective of the NGN is to deliver services across heterogeneous platforms. As a result of this objective, the NGN is a good context on which to base the design of the teleworking service. The teleworking service can be designed as a service that can be used independent of the network platform, and consist of services that combine voice, data and video on the same platform. The teleworking service thus becomes an enhanced service as compared to already existing ones.

The OSA / Parlay architecture allows the teleworking service to be designed in conformance to the NGN concepts. It also allows not only the telcos to develop the teleworking service but opens up the network for the service providers to develop the teleworking service. Teleworking, NGN and OSA / Parlay are discussed in this introduction for a good understanding of the teleworking service design.

1.1 Teleworking

Teleworking is the generic term for work that can be done at a distance. Teleworking is the use of information and communication technology to enable people in different geographical locations work effectively together. Teleworking makes possible a more flexible design of working time and place. Teleworking takes the work to the people and not the people to work [1], and is important in today's context of ever-rising fuel prices.

Many activities, which previously have required physical presence and direct interaction among workers, can now be performed in a distributed fashion, thanks to the help of advanced information technologies such as interactive multimedia services and new communication facilities [2]. Teleworking service can be used for activities such as banking, insurance, telecommunications, computing, accounting, designing, sales, teaching, research, and communication.

There are different forms of teleworking service. The above-mentioned activities can be used in any of the different forms of teleworking.

1.1.1 Forms of Teleworking

Teleworking comes in different forms. Working from home is the basic and most common form of teleworking. The different forms are discussed below [3]:

1. *Home-based teleworking*: Home-based teleworking is when employees work at home instead of their usual work place thereby avoiding commuting from home to the office or the customers' premises.
2. *Mobile teleworking*: Mobile teleworking occurs when workers use ICT to enable them deliver a range of services and capabilities that previously would have involved office-based staff or physical visits to the company's offices.
3. *Telecentres*: Telecentres may be used to provide teleworking local office facilities for people who prefer not to work at home but wish to avoid the cost, time and inconvenience of commuting all the way to the work place.
4. *Mixed mode*: Mixed mode is a combination of teleworking and normal commuting. Workers work from home for a certain number of days in a week.

The American Interactive Consumer Survey conducted by the Dieringer Research Group in Milwaukee shows the number of employed individuals who work from home during business hours at least one day per month has increased by nearly 40% since 2001 [4]. The results show that 16.8 million employees worked from home at least one day per month two years ago, while the number climbed to 23.5 million this year. The number of self-employed people working from home at least one day per month in 2001 was 19.9 million with this year's number reaching 23.4 million [4]. At present, teleworking is most advanced in the United States. However, only 3% of European employees are engaged in teleworking along with an estimated 20 million worldwide. Predictions are that by the year 2006, the number could increase tenfold [5].

The numbers specified in these statistics are increasing because there are many benefits to be derived from teleworking.

1.1.2 Benefits of Teleworking

Teleworking offers benefits to the employer, employee and to the community as a whole.

1. **Benefits to the Employer**

    - Reduced Absenteeism: Absenteeism can be due to sickness. Due to the flexible working hours and work place, there is a reduction in absenteeism. Some workers are used to taking the whole day off after attending a social function even though there is still enough time to go back to work. People involved in teleworking have been known to go back to work after such social functions.
    - Increased Productivity: There is increased productivity as a result of reduced absenteeism and less stress for workers.
    - Cost Savings: There is a reduction in office space costs because less people work in the traditional work place.
    - Staff Retainment: Employers can retain their staff who are not willing to relocate. Despite a change in office location, employees can still be retained.

2. **Benefits to the Employee**

    - Reduced stress: The elimination of long and difficult commutes translates into less stress for the employees and their families.
    - Better work-family balance: Workers get to spend sufficient time with their families and also at work, thus establishing a better balance between work and family.
    - Greater flexibility of work and private life: For instance, workers can decide to do their shopping at off-peak times and work at night or peak shopping periods.
    - Improved job satisfaction: There is improved job satisfaction because the workers work at flexible times. This in turn leads to increased productivity.
    - Reduced transportation cost: Transportation cost is greatly reduced as a result of less commuting.

3. **Benefits to the Community**

    - Increased job opportunity: Disabled people can still be employed even though they cannot commute to work. This increases job opportunities in the community.

Considering the various benefits of teleworking, it becomes necessary to design a teleworking service that is reliable, readily available, user friendly and meets the standards of the evolving open network architectures.

1.2 Next Generation Networks (NGN)

The evolution of networks and services towards the NGN is a major trend in telecoms, and of great interest to the telecommunications market. New entrant operators are introducing new network architectures, technologies and services to the market. Incumbent operators are enhancing their networks and services to meet changes in the market demand, which is a result of the evolving NGN. Third-party service providers use service differentiation to penetrate the market.

The NGN seamlessly blends the Public Switched Telephone Network (PSTN) and the Public Switched Data Network (PSDN), creating a single multi-service network [6]. The NGN delivers multi-services across heterogeneous network infrastructures. The NGN is a multi-service bearer network capable of supporting multi-party, multimedia, real-time and information services [6]. The NGN is based on a progressive movement towards an end-to-end fully Internet Protocol (IP) network to adapt to major trends: convergence, flexible network evolution, distribution of network intelligence and openness to third-party services [7]. For this network evolution to be successful, the different interests of the major role players of the telecommunications market must be integrated.

The NGN role players and their interests are [8]:

- **Network operators** who need to manage the transition of their service business to the NGN.
- **Service providers** who see significant market opportunity for services that exploit the new possibilities inherent in the convergence of the data and voice technologies.
- **Application developers** who see a whole range of new business opportunities that were previously not technically feasible or too costly for small firms.
- **End-Users** who expect to see new services that provide an integrated interface and consistent access to all their communications and information needs.
- **Network equipment providers** who wish to provide a sound migration to the evolving service architecture and also need to ensure interoperability between the data and voice networks.

In the NGN, intelligence is pushed to the edge of the network. This results in a distributed network infrastructure. The move of the telecoms sector to the NGN is as a result of the following favourable and motivating factors. These factors are [7]:

- major structural changes to the telecoms market e.g. deregulation of the telecoms markets, network optimisation and cost reduction.
- major changes in services and uses e.g. explosion of data services, multimedia, mobility and accessibility has led to the need for operators and service providers to develop new markets.
- major technological evolutions brought about with the development of very high-speed access transport networks, the generalisation and evolution of IP in favour of different levels of Quality of Service (QoS).

The NGN converges voice and data and video into one network. This network transformation has allowed applications to be pushed closer to the end-user, thereby reducing overall cost and at the same time, enhancing system flexibility and functionality. Services are developed independent of the transport layer i.e. the service providers need not know anything about the transportation technology. Applications development is all that needs to be done by the service providers. Figure 1.1 shows an overview of the NGN architecture with the service layer separated from the transport layer.

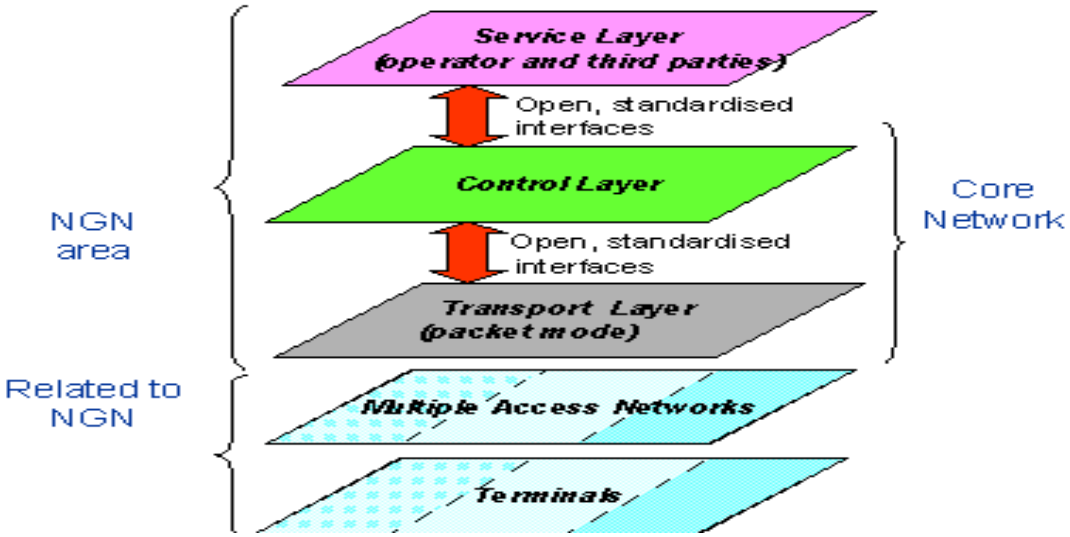

Figure 1.1: Architecture of NGN
[7]

1.2.1 Characteristics of the NGN

The characteristics of the NGN are that it is:

- a unique and shared core network for all types of access and services. It provides support for multiple applications, which are adaptable to the users and the varied capacities of access networks and terminals.
- a core network architecture that is divided into three layers: transport, control and services (application). The NGN separates the application layer from the transport layer. The application layer is where services are developed by the trusted third-parties while the transport layer, which deals with transport is managed by the telco.
- a network that provides open and standardised interfaces between each layer. The interfaces allow third-party providers to develop and create services independent of the network transport technology. These standard interfaces enable easy new service creation through their flexibility. High extensibility through open architecture design implies an easy introduction of future technologies.
- based on the packet mode of transport.
- able to support currently existing analogue and digital network standards, service elements, thereby saving cost.
- highly scalable.
- capable of producing increased revenues via shortening time to market, thus reducing production / service creation costs.

In summary, the NGN concept is designed to converge all types of communication networks. The missing link between the different role players' interests is a common set of languages to communicate with each other. The open OSA / Parlay API provides this missing link and enables all NGN role players to communicate with each other or to exchange data in a very secure and safe manner. The OSA / Parlay architecture consists of interfaces that separate the application layer from the transport layer. The OSA / Parlay API opens up the network in a secure and reliable manner.

1.3 OSA / Parlay Architecture

Over the years, networks have been closed, but today, there is a move towards the open networks. A *closed network* is a network in which service providers outside the telco cannot

control the network resources, i.e. cannot provide or develop services. The Internet for instance, is a closed network in that no services other than packet transport can be leveraged out of the bearer network.

An *open network* permits a service provider other than the telco to create and offer services, which make use of resources within the network, e.g. voice connections. An open network is achieved by accessing the network service control through an open, standard and secure interface with a defined Applications Programming Interface (API) [9]. An *API* is a set of software functions used by an application program as a means for providing access to a system's (i.e. operating, communications) capabilities [10]. An API is an interface to existing network technologies and signaling systems. It hides the complexity of network and signalling. The API is the key to cost-effective service-provision. The objectives of the API paradigms are: rapid service creation and a new business model. The API exposes control of many network capabilities to applications residing outside the telco's network, at a high level of abstraction [11]. A dominant API approach is the OSA / Parlay API.

The OSA / Parlay open API specifications are provided by the Parlay group, which consists of major telecommunication network suppliers, major IT vendors and the telcos. The Parlay group was formed in April 1998 and now consists of all major players in the communications industry including Ericsson and AT&T [8]. The goal of the Parlay group is to expand the Parlay API specifications, thus accelerating the convergence of wireless networks, IP-based networks and the PSTN [8].

The OSA / Parlay is an architecture that opens up the telecommunication network functionality for application development. The OSA / Parlay architecture empowers application service providers, independent software providers and other developers in the ICT industry to generate new applications that add value to the functionality available in public and private communication networks [12]. The OSA / Parlay API specifications enable a new generation of dynamic telecommunications applications created and maintained by the networks' customers themselves using their own private data. It is a secure, technology and network independent API. It enables the telco to directly pass on the technical functionality to trusted third-parties in a safe and controlled manner, while hiding the network signaling complexity from the developer as required for the NGN. OSA / Parlay business model defines the role players that are participating to deliver advanced telecommunications and information services to the end-user. The OSA / Parlay business model and the role players are shown in Figure 1.2.

The OSA/Parlay business role players are the:

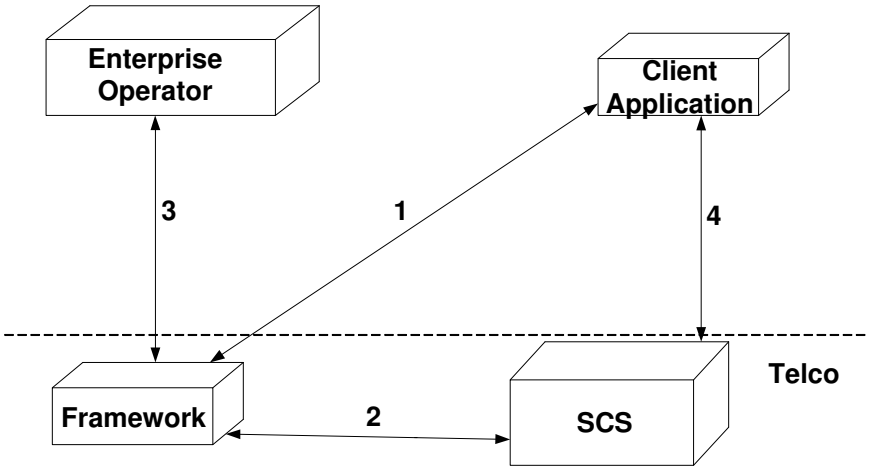

Figure 1.2: OSA/Parlay Business Role Players

- Telco: The telco offers basic connectivity, connection and service control. The interfaces in the telco domain must be secure, manageable and well defined. The framework is in the telco's domain.
- Enterprise Operator: The enterprise operator offers and uses applications outside the telco domain. Such an application is termed the client application, i.e., the client of the telco.
- Client Application: The client application is the consumer of the services supported by the APIs. An application that accesses distributed objects provided by a server application [13].

In Figure 1.2, number 1 represents the interactions between the client application and the framework. Number 2 represents the interactions between the framework and the Service Capability Server (SCS). Interactions between the framework and the enterprise operator are represented by number 3 while number 4 denotes the interactions between the client application and the SCS. These interactions are discussed later.

The OSA / Parlay architecture is made up of two basic interfaces, these are the framework interfaces and the service interfaces. The *service interfaces* offer applications access to a range of network capabilities and information. The *framework interfaces* offer framework capabilities to applications. The OSA / Parlay gateway contains the framework and SCS. The SCS contains the Service Capability Features (SCFs). The SCFs are made up of interfaces and they are offered to the applications through the API in addition to containing the network capabilities. The applications are deployed on the application servers and can be on any network-platform. The applications use SCFs in service provision. API specifications provide callback interfaces in the service provider domain. A callback serves as

a contact point for responses, notifications and errors in a particular domain for an object residing in another domain . Figure 1.3 shows that these are the entities that make up the OSA / Parlay API. The SCFs and the framework are in the telco domain. The applications are in the service provider domain. There can be more than one Parlay gateway in the telco domain. The telco opens up its network to allow the service provider to access the network capabilities.

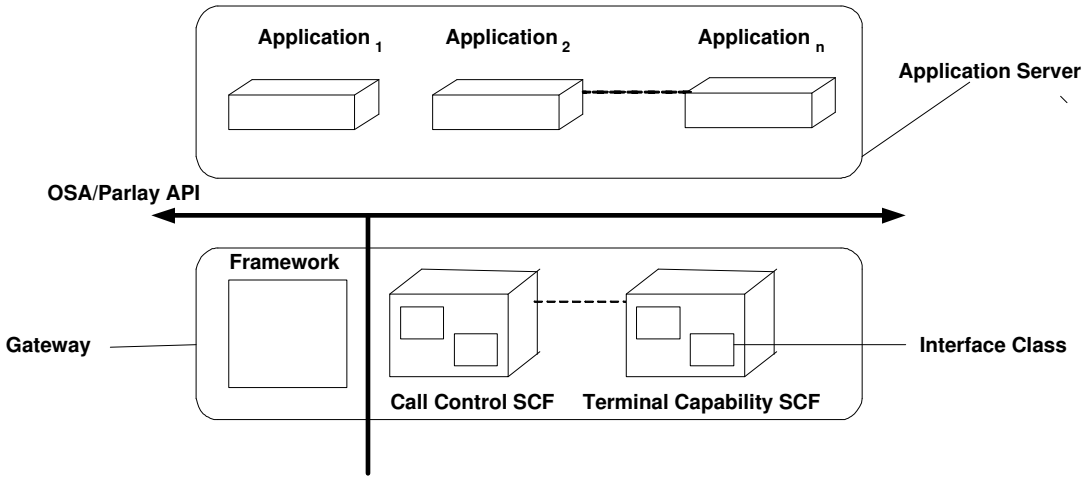

Figure 1.3: Overview of the OSA/Parlay Architecture

1.3.1 OSA/Parlay Entities

1. **The Framework**

   The *framework* provides capabilities necessary for the service interfaces to be secure and manageable. The framework enables discovery and integration of new service capability features. The framework provides controlled access to the SCFs. The interfaces that are implemented by the framework are referred to as the *framework interfaces*. The corresponding interfaces that must be implemented by the application are denoted as application interfaces. Interfaces of the framework are categorised according to the interaction between the framework and other OSA / Parlay entities / business parties.

   - **Framework and Client Application**

     The interactions between the framework and client application is marked 1, in Figure 1.2. The interfaces involved in the interaction between the framework and the client application are the *trust and security management interfaces*. The

trust and security management interfaces support initial contact of the application with the framework. They provide the first point of contact for an application to access the framework. This interface also supports:

(a) **Authentication** of the client application.

(b) **Authorisation** permits an already authenticated client to invoke operations on the framework.

(c) **Service discovery** helps in discovering services and registered SCS available in the gateway. It gives information about the availability and properties / description of a service instance.

(d) **Access control to SCFs** permits access of authorised clients to SCFs.

- **Framework and SCS**

  Number 2 in Figure 1.2 shows the interaction between the framework and the SCS. The interactions are basically for registration of the SCS.

  (a) **The service registration interfaces** support the registration of an SCS with the framework. When registered, the framework can give information on the availability of the SCS and its properties. A service has to be registered before it can be discovered, subscribed to or accessed. Services are registered against a particular service type. The framework maintains a repository of service types and registered services.

- **Framework and Enterprise Operator**

  Number 3 in Figure 1.2 depicts the interactions between the framework and enterprise operator.

  (a) **The service subscription interfaces** support the interaction between the framework and the enterprise operator. The service subscription interfaces support the subscription of the client application to a service. Subscription is at times necessary before the client application can access or use a service. In the Parlay business model, the enterprise operators act in the role of the subscriber who subscribes to a service. The client application is the end-user, and the framework acts as the retailer of the service.

Other framework interfaces are for:

- **Event notification**: *Event notification* is a mechanism that is used to notify the application of generic service related events that have occurred.
- **Integrity management** includes:

  (a) Load management

  (b) heartbeat mechanism

  (c) Operations, Administration and Maintenance (OAM).

The *load manager* allows the load to be distributed across multiple machines according to a load management policy. The load management policy is related to the QoS level to which the application is subscribed [8].

The *interface for heartbeat management* allows the initialisation of a heartbeat supervision of the application.

The *OAM interface* is used to query the system time and date. The date and time can be synchronised by the framework and the application.

The interactions between the client application and the SCS as shown by number 4 in Figure 1.2 involve the sending of requests to the different SCFs and receiving responses from the SCFs.

2. **Application Server**

   The application server contains the different applications that are supported by the OSA / Parlay gateway.

3. **Service Capability Server (SCS)**

   The SCS provides the applications with all the functionality, called network SCFs, required to create services. The SCS is implemented in the underlying networks and there is at least one SCS in each network. The SCS communicates with network entities like the Home Location Register (HLR) and the Mobile Switching Centre (MSC). The SCFs that make up the SCS are:

   The **Call control SCF** supports the routing and creation of calls between leg objects.

   - **Generic call control SCF** provides the basic two party call control service for the API. It allows calls to be instantiated from the network and routed through the network. A leg can be attached (media or bearer channels related to the legs are connected to the media or bearer channels of the other leg that is attached to the call) or detached from the call. The application can request to be notified of calls that meet certain criteria or the application can initiate a call. These are two ways by which an application can get the control of a call.
   - **Multiparty call control SCF** enhances the functionality of the generic call control service with leg management. It allows for more than two legs to be connected simultaneously. The number of legs connected at the same time is service specific.
   - **Multimedia call control SCF** enhances the functionality of the multiparty call control with multimedia capabilities. A media channel is introduced, which is a unidirectional media stream that is associated with a call leg.
   - **Conference call control** adds the capability of manipulating sub-conferences within a conference to the application. A sub-conference defines the grouping

of legs within the overall conference call [8]. The conference call control service adds support for the reservation of resources needed for conferencing.

The **User interaction SCF** is used by applications to interact with end-users. The generic user interaction service interface handles the sending and receiving of information from the end-user. The call user interaction service interface provides functions to send information to and receive information from the end-user to whom a call leg is connected.

The **Mobility SCF** is used to implement applications with a close relationship to mobility. The services in the set are User location, User status, User location camel and User location emergency.

- **User location** provides a generic geographical location service. It allows applications to get the geographical location and status of fixed, mobile and IP based network users.
- **User location Camel SCF** provides location information based on network related information. Using the User location camel functions, an application can request for the Visitor Location Register (VLR) number, cell global identification and other mobile related network information.
- **User location emergency SCF** service is used in the event of an emergency, for locating the caller automatically.
- **User status SCF** service provides a general user status service, which allows applications to obtain the status of a fixed, mobile and IP based network user. The user status are: reachable, not reachable and busy.

The **Terminal capability SCF** enables applications to retrieve the terminal capabilities of a specific terminal. The retrieval of terminal capabilities that are caused by some triggering in the network or terminal is also supported. The availability of a terminal can also be retrieved using the terminal capability SCF.

The **Data session control SCF** provides a mechanism that is used to control the establishment of new data sessions and existing data sessions.

The **Generic messaging SCF** is used by applications to send, store and receive messages. The messaging system is assumed to consist of mailboxes, folders and messages. The mailbox is the application's main entry point to the messaging system. The folders make up the mailbox and the messages are contained in the folders.

The **Accounting Management SCF** supports methods for monitoring and managing accounts.

The **Charging SCF** is used by applications to charge for service usage.

The **Policy management SCF** allows the service provider and telco to define policies governing the usage of resources during service provisioning.

The **Presence and Availability (PAM) SCF** facilitates the development of a rich set of applications and services that span over multiple communication systems. It also provides the end-user with increased flexibility and control for managing voice and data communications.

Other considerations of the OSA / Parlay API include security and QoS. Security is essential when applications from different network domains need to interact, use SCFs or applications from other domains. Security is achieved via authentication between domains on API level. The whole OSA / Parlay architecture offers an end-to-end security. Besides this, the API is open, yet secure. Service agreements can be used to specify and ensure QoS. QoS can be specified by the application.

OSA / Parlay uses an object-oriented technique, like CORBA, thus making technology and platform independence possible.

1.4 Objectives of the Research

A teleworking service offers a flexible working time and place, thereby greatly increasing productivity. There are some existing forms of services that make up the teleworking service over the Internet. These services include the e-mail, conferencing and Voice Over IP (VoIP). These existing forms of teleworking service do not offer a guaranteed QoS or high enough security and the services are in isolated forms and not integrated into an easy-to-use package.

This research examines the *high level* design of teleworking service with access to telecommunication and data services based on the defined concepts of the NGN. The objective of this research is to develop an integrated teleworking service that is composed of simpler services, in a distributed processing environment. These simpler services include e-mail, conferencing, shared whiteboards, joint document editing, conventional phone calls, mobile communications, multimedia presentations, data transfers, video on demand and user profile managements. The teleworking service must be designed so that it is available for subscription by corporate bodies and individuals. The design of the teleworking service logic must be developed to demonstrate two significant aspects, which are communication between teleworkers in different providers' domains via the interworking of the providers' services based on an agreement between the providers for service and resource usage. Also the construction of new services must be possible by reusing one or more existing services.

These two aspects are referred to as *service federation* and *service composition* respectively. A secure way of achieving service federation and service composition is needed. The teleworking service must be developed in a way that ensures that the underlying networking technologies are hidden from the service providers and the service consumers.

This teleworking service is to be designed to be a service that supports:

- integration of the mentioned simpler services such as email, conferencing and telephony;
- interaction between end users in the same or different service provider domains;
- QoS specification;
- high security.

The design of this teleworking service exploits federation, which is needed in the actual implementation to realise service federation and service composition.

The OSA / Parlay architecture will be used to design the teleworking service. Of the NGN architectures, only TINA addresses the issues of service federation and service composition. Ideas will therefore be taken from the TINA architecture.

1.5 Report Outline

The rest of this report is organised as follows:

Chapter 2 gives a background to the proposed implementation strategy for service federation, that is, the framework federation. The trader federation approach to service composition, is also discussed in this chapter. In Chapter 3, we look at the Reference Model for Open Distributed Programming (RM-ODP) enterprise viewpoint, focusing on the high level requirements of the teleworking service. We identify the enterprise objects and the roles they play to make the teleworking service work effectively. Chapter 4 examines the RM-ODP information viewpoint. The flow of information with respect to the general working of the teleworking service, framework federation and trader federation are discussed. In Chapter 5, we discuss the application of the OSA / Parlay architecture in terms of computational objects, interfaces and methods. We also look at the newly introduced framework interfaces. An application example is given for service federation and composition in Chapter 6. The audio conference and shared whiteboard services are used for service composition. Finally, we present the conclusions in Chapter 7, and give recommendations for further work.

Chapter 2

Service Federation and Service Composition

Service federation and *service composition* are two basic aspects of a teleworking service. Teleworkers need to interact with each other and the interactions can be within the same or different service provider domains. Interactions across different service provider domains is vital as communications across different domains often occur. If the interaction involves different domains, the service providers have to interwork based on a contract to realise this interdomain interaction. Interactions across different service provider domains is vital as communications across different domains often occur. Teleworkers should also be able to access services in a service provider's domain that is different from their resident domain.

The integration of simpler services is necessary for a good teleworking service. It should be possible to combine one or more services together to create a compound service. The above mentioned basic aspects of teleworking are referred to as service federation and service composition respectively.

The implementations of service federation and service composition exist in some open network architecture environments such as the TINA environment, but not in the OSA / Parlay environment. There is the need to look for a suitable approach to these two essential aspects of teleworking within the OSA / Parlay environment that will achieve high security and required QoS.

According to the TINA consortium glossary, *federation* is an organisational structure involving two or more autonomous stakeholders, in which the members have an agreement on how they will interwork with each other including the extent to which the resource of one member can be shared by other members [14]. From the definition of federation, *service federation* can then be said to be an organisational structure that permits the interworking of services owned by autonomous service providers based on an agreement between the service providers for resource and domain usage.

Composition is an action or the result of realising a certain object or service from a set of basic or simpler objects or services [14]. *Service composition* is the construction of new services by reusing one or more existing service components or services within the same service instance [14]. The constructed new service is referred to as a *compound service*. The reused services can be provided by the same or different service providers.

2.1 Service Federation

A *Service federation* can be seen as different providers' services interworking. An end-user communicating with another end-user in a different provider's domain is a form of service federation. An end-user using the services provided by a service provider different from its own service provider is another form of service federation. The service providers have to interwork to realise communication and the use of services across their domains. One may think of service federation as being analogous to roaming. This is true to an extent but the difference is that in service federation, the end-user uses services in another domain while still residing in its provider's domain. The extent of resources usage is agreed upon for the successful interworking of the services. Roaming deals with the same user accessing his service in other providers' domains. Figure 2.1 shows a view of service federation. Adopting the TINA consortium glossary definition of service federation, a federation contract must exist between service providers before a service federation can occur. The federation contract is an agreement between the service providers as to domain and resource usage.

In the OSA / Parlay context, service federation allows the use of SCFs and services across different service providers' domains. Service federation supports service providers cooperating with each other to ensure low cost and worldwide customised and transparent service provisioning. A service provider can decide not to offer an SCF but use the SCF of another service provider to ensure low cost.

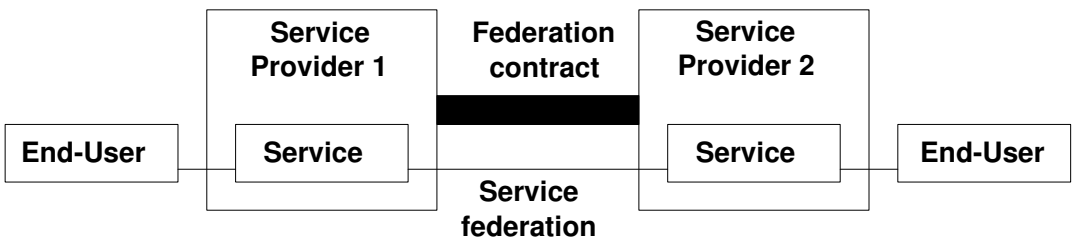

Figure 2.1: Service Federation
[15]

A scenario for service federation is that of a user in a fixed network wishing to communicate with another user in a mobile network. The fixed phone service provider needs to communicate with the mobile phone service provider for access to the gateway MSC of the mobile network, to route the call. The location and terminal capability of the terminal in the mobile network can be retrieved from the SCFs in the domain of the mobile network service provider. This is achieved if a federation agreement referred to as party interconnect exists between both service providers.

For seamless communication with a high QoS, the originating and destination terminals must possess compatible terminal properties and SCF supports. The originating terminal needs to be aware of the terminal information of the destination terminal such as the terminal capability, user availability, type of services and interfaces supported on the destination terminal. Other information are user status and geographical location. When the parties are in different service providers' domains, retrieving the needed terminal information using the OSA / Parlay architecture, leads to another type of federation, which is termed framework federation.

2.2 Framework Federation

A *framework federation* is the interworking of two or more Parlay frameworks owned by different providers to achieve service federation. There is a federation agreement between the service providers as to framework usage. Framework federation allows for secure, seamless and reliable communication across service providers' domains. Framework federation makes service federation possible in the OSA / Parlay environment. Framework federation can also be said to mean integration of Parlay gateways because the frameworks reside in the gateways. A Parlay gateway is made up of a framework and SCS. Framework federation enables an application to gain access to a gateway of another service provider and use SCFs residing in that gateway. The OSA / Parlay frameworks are made accessible to the service providers by the telco. This means that each service provider is associated with one or more gateways, hence one or more frameworks. The application represents the service logic of the teleworking service that handles the requests of the end-user. The requirements for framework federation are:

- Integration of Parlay gateways of different service providers. As a result of each framework being contained in a gateway, framework federation requires the integration of gateways, i.e., communication between gateways. These gateways are the gateways in which each of these frameworks resides.

- Interoperability of geographically distributed Parlay gateways. Gateways, which are in different geographical locations, can still communicate and interoperate with each other based on framework federation.

2.2.1 Existing Implementation Strategies for Framework Federation

Presently, the TINA architecture supports service federation and service composition. TINA realised service federation and service composition via the use of computational objects like the Composition User Service Session Manager (CompUSM). There is a standard-based approach to framework federation within the OSA / Parlay environment but this approach is not so straight-forward [16]. A temporary approach, known as the Proxy manager approach [16] also exists.

Standard-Based Approach

The standard-based approach is shown in Figure 2.2. It is called the standard-based approach because it adheres to the OSA / Parlay standards. The *visiting gateway* consists of the visiting framework (service provider) and resides in the same domain as the application that wishes to access a gateway and use SCFs from another provider's domain. The application in the same domain as the visiting gateway has access to SCFs residing in its provider's domain but not SCFs in another provider's domain. The application that pertains to the visiting gateway can be referred to as the originating terminal service logic.

The *visited gateway* consists of the visited framework and other related SCFs. The destination terminal and gateway reside in this provider's domain. The visited gateway can be referred to as the host gateway. The application in the visiting gateway accesses the visited gateway for call routing and also uses SCFs that are in the visited gateway.

The steps to the working of the standard-based approach are as follows:

1. register an SCS from the visited gateway in the visiting framework,
2. restrict usage of the SCS (setup service properties) to a specific application,
3. announce availability of the new SCS.

The application in the visiting gateway can now discover and use SCS supported by the visited gateway.

The requirements for the standard approach are [16]:

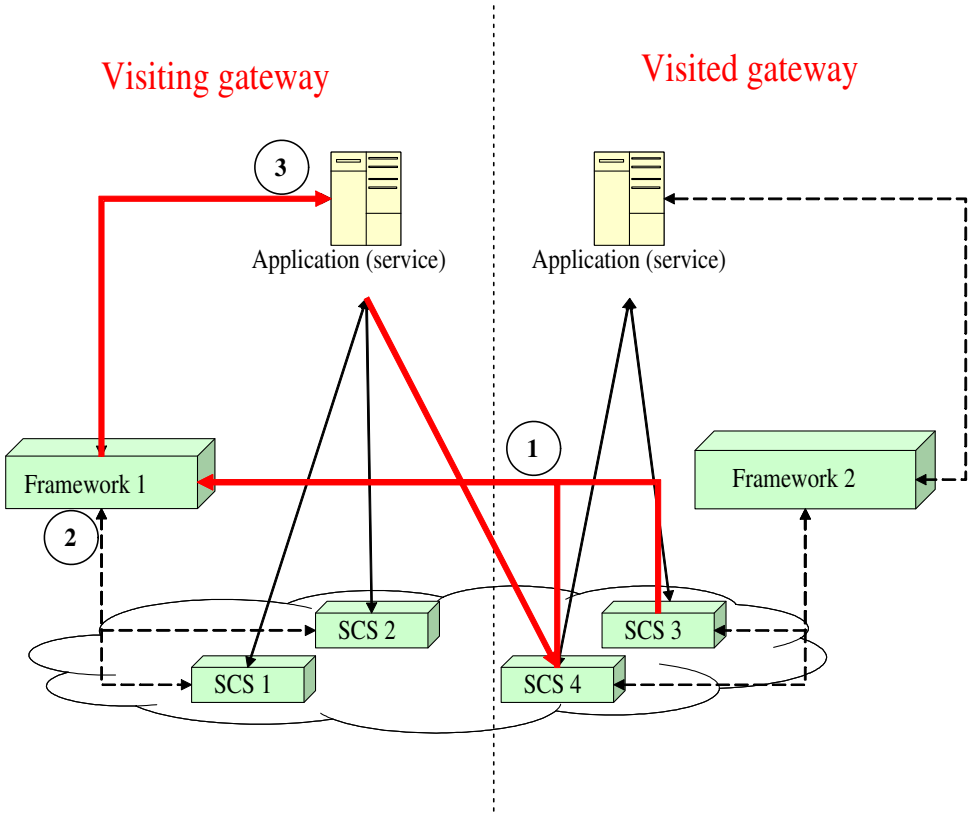

Figure 2.2: Standard-based Approach
[16]

1. SCS shall support:
   - standard registration procedures and
   - simultaneous registration in multiple frameworks
2. SCS registration with a new framework while still registered with original framework must be possible.
3. SCS shall support usage based on service properties. The values of the service properties are given to an SCF by a framework during the process of assigning a service manager to an application [16].
4. The framework shall support:
   - standard procedure of the service registration and
   - service properties

The disadvantages of the standard approach are that:

1. not all service providers implement all of the Parlay features required for implementation of the framework federation approach. There will be a limitation when an SCS wishes to register with another framework that does not support the Parlay features of its framework.

2. the standard-based approach does not involve the visited framework in the federation. The inclusion of the visited framework can help to enhance security via authentication.

3. the Parlay specifications do not provide a standard set of service properties that allow usage control across SCS and frameworks of different service providers.

Proxy Manager Approach

The second approach to framework federation is the use of an object, which is called a proxy

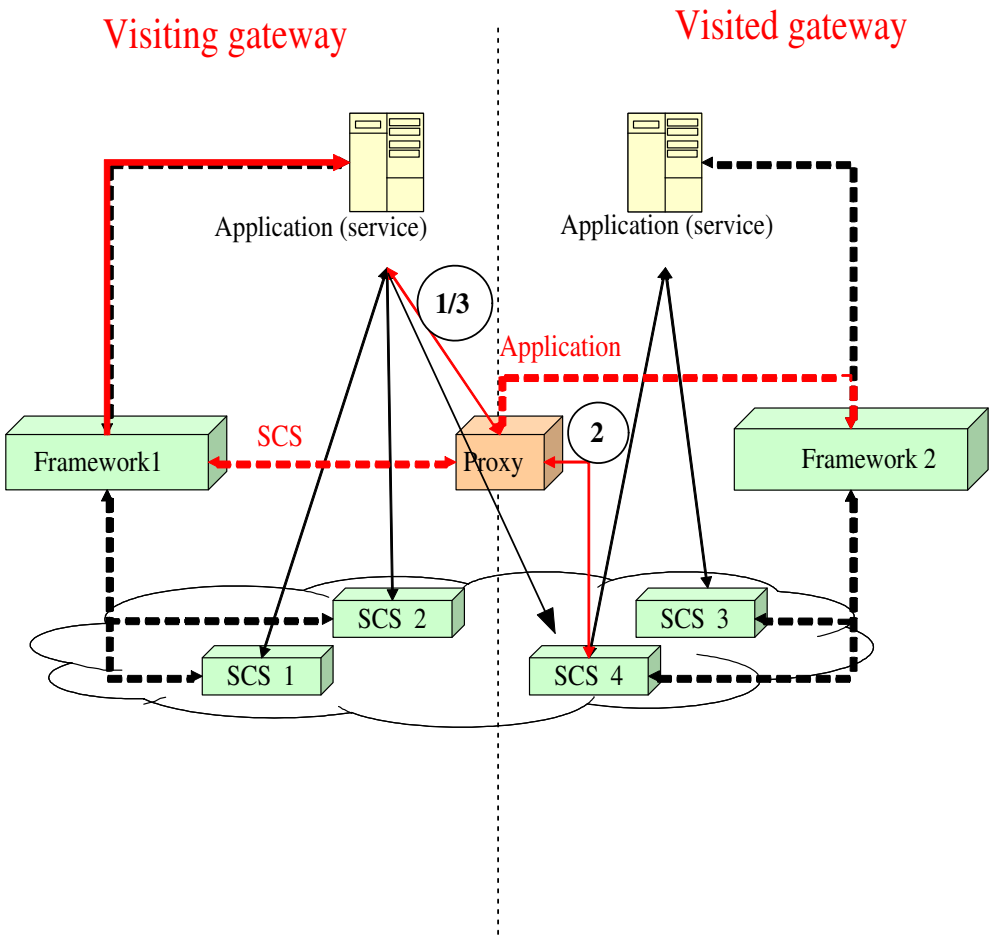

Figure 2.3: Proxy Manager Approach
[16]

manager. The proxy manager serves as a computational object between the application of the visiting framework and the SCS of the visited framework as shown in Figure 2.3. As the name implies, the proxy manager functions like the regular proxy server. The proxy manager is an intermediary entity that acts as both an application and a SCS. The proxy manager ensures that a request is passed from the application in the visiting gateway to the SCS in the visited gateway, that further processes the request.

To the application in the visiting gateway, the proxy manager looks like an SCF, i.e., the application makes requests to the SCFs via the proxy manager. The application is authenticated and authorised by the proxy manager on behalf of the visiting and visited frameworks. The proxy manager appears to be an application to the SCFs in the visited gateway, which means that the proxy manager can directly interact with the SCFs. The application accesses the SCFs in the visited gateway through the proxy manager and this occurs as follows:

1. The application makes requests to the proxy manager for the usage of specified SCFs.
2. The proxy manager verifies that the application is authorised to use the specified SCFs and gets the interface reference from the visited framework. The proxy manager passes the application's request to the appropriate SCFs.
3. The reports from the SCFs are returned to the application by the proxy manager.

A requirement of the proxy manager approach is that the proxy manager must be able to make requests across multiple gateways.

Apart from being an intermediary object between both frameworks, the proxy manager increases security and provides load balancing and fault-tolerance. A disadvantage of the proxy manager approach is that it is a temporary approach.

As a result of the limitations of the OSA / Parlay specifications to support the standard-based approach and the temporary nature of the proxy manager, we propose a new approach to framework federation. This proposed approach is a potential addition to the OSA / Parlay specifications.

2.3 Proposed Approach to Framework Federation

The proposed approach to framework federation allows both the visited and visiting frameworks to directly communicate with each other to achieve framework federation. Within the same provider's domain, the service provider has access to one or more gateways that

each consist of a framework and SCFs. Each SCF is registered with the framework, thus the framework knows the properties and descriptions of all SCFs registered with it and the interface reference for accessing the SCFs.

When dealing with more than one service provider's domain, the visiting framework gets all the required information about how to access the needed SCFs and terminals, from the visited framework. The visiting framework returns reports, which contain information about the location of the required terminal and SCFs, to the application. By locating and getting access to the SCFs, call routing can occur and the end-user is able to use the SCFs in the visited gateway.

To fully understand the requirements for this proposed approach to framework federation, the registration of SCS with the framework has to be well understood. The steps to registering and accessing SCS are as follows:

1. A new SCS is available to the gateway.
2. The new SCS is added to the gateway and the SCS authenticates itself with the gateway.
3. The SCS requests for the reference to the framework's registration interface so as to know where to register its SCFs.
4. The SCS supplies its interface reference to the framework. This is the reference to the factory interface.
5. The framework is aware of the availability of the SCS and announces the available SCS. Announcing the SCS availability does not make the SCS accessible by the application.
6. The application makes a request to the framework for the reference to the service discovery interface, to discover the available SCS.
7. The application invokes operations on the discovery interface and selects the SCS.
8. The framework creates the service manager in the gateway via the interface registered by the SCS in step 4.
9. The service manager reference is returned to the framework.
10. The framework passes the service manager interface to the application. The application creates a callback interface in the application to receive responses and notifications from the service manager in the gateway.

Having established how an SCS is registered and how applications gain access to it, we now discuss the steps to achieving service federation using framework federation. Figure 2.4 shows the diagram of the proposed approach to framework federation.

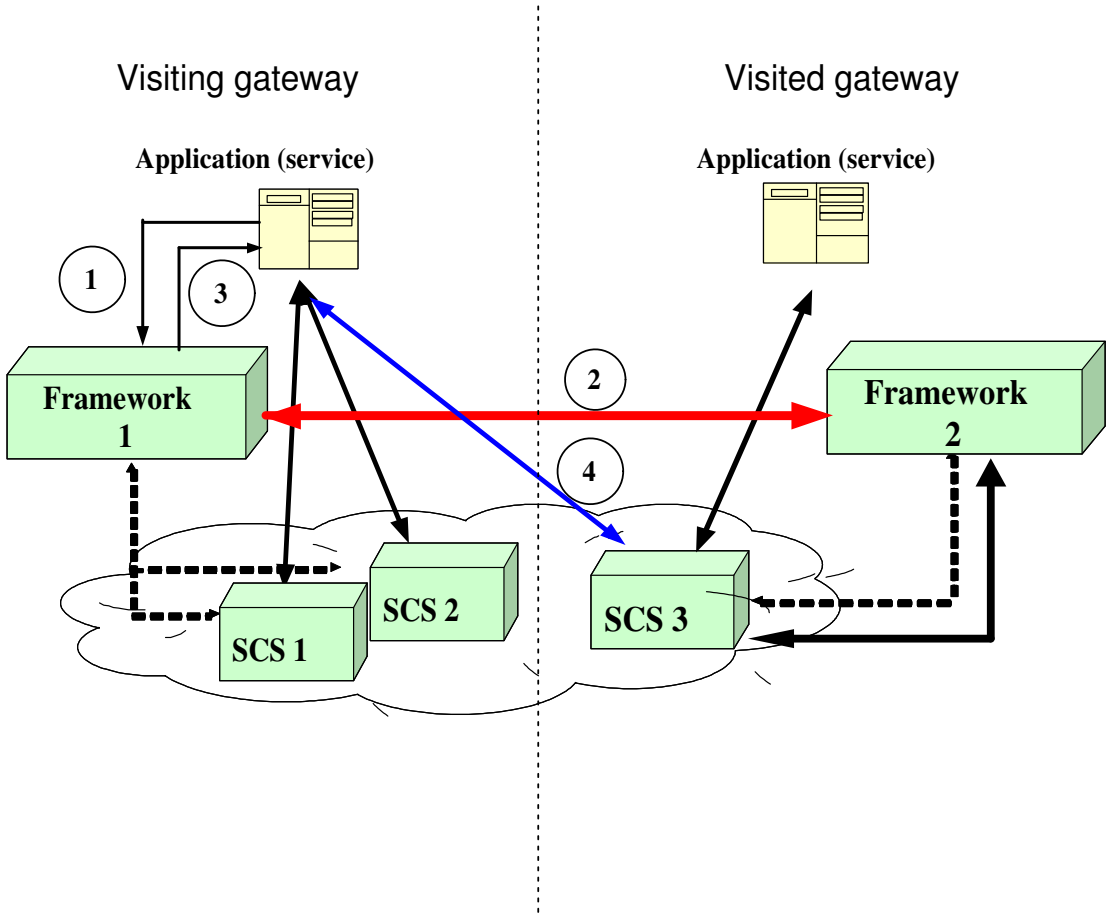

Figure 2.4: Proposed Approach

1. The application in the visiting gateway's domain makes a request to the visiting framework to use SCFs in the visited gateway. The new framework interfaces, fully described in Chapter 5, are required to support inter-framework operations.

2. The visiting framework passes the application's request to the visited gateway and gets permission for the application to access interfaces on the visited framework.

3. The visiting framework returns the permission report to the application.

4. The application, via the references, discovers and locates the required SCS and accesses the required SCFs. The visited framework creates the service manager of the selected SCF if necessary (not all SCFs have a separate object as the service manager). The service manager reference is returned to the application and the application creates callback objects for responses.

The present OSA / Parlay specifications do not yet support this proposed approach. New framework interfaces are proposed for the realisation of this proposed approach. The new framework interfaces are: Request Interface and Network Information Interface. An advantage of this proposed approach is the reuse of already existing OSA / Parlay framework operations e.g. for framework to framework authentication. This approach to framework federation is a potential addition to the OSA / Parlay specifications.

2.3.1 New Framework Interfaces

1. **Request Interface**: The *request interface* allows the application to send requests to the framework other than the authentication and access requests. Such a request is an Invitation request or a request to use SCFs in another provider's domain. The framework, via this interface, returns the necessary reports that will enable the application to locate and gain access to the required SCFs.

2. **Network Information Interface**: The *network information interface* supports the passing of requests between frameworks. The visiting framework passes requests made by the application to the visited framework through this interface. The visited framework returns reports that contain permission for the application to access its interfaces. This interface can also be used by the visiting framework to discover the QoS level supported by the visited framework and other network related information like QoS negotiation.

The detailed definition of these new framework interfaces is given in Chapter 5. The existing trust and security management framework interfaces can be used for framework to framework authentication, authorisation and access requests. This involves the authentication of the visiting framework by the visited framework or vice versa and also the request for access to other framework interfaces. These new framework interfaces will support the proposed approach to framework federation.

2.4 Service Composition

Another important aspect of this teleworking service is service composition. *Service composition* is the creation of a service instance by reusing one or more other services or service components. Service composition leads to the realisation of a compound service, which is a service created by reusing one or more services or service components. The end-user can use another service within the same service session, while still running a service. Service

composition can be within the same service provider domain or across different service providers' domains. For instance, a group of users of the teleworking service may be using a chat service and decide that they can benefit from using a shared whiteboard service to help in viewing some documents. The start of this shared whiteboard service while still running the chat service is an example of service composition. There are two types of service composition:

1. **Static service composition**: Service composition is carried out before the service session starts.
2. **Dynamic service composition**: Service composition is carried out after the start of the service session.

Service composition is closely related to service federation. The difference is that service federation is concerned with provision of the same services to users associated with different service providers while service composition is for a user associated with a single service provider who is willing to use a service provided by another service provider within a service instance as shown in Figure 2.5. A federation contract must exist between the service providers when more than one service provider domains are involved. Service composition aims to be as flexible and open as possible to allow different types of combinations of services [15]. A trading service is needed to search for the required services.

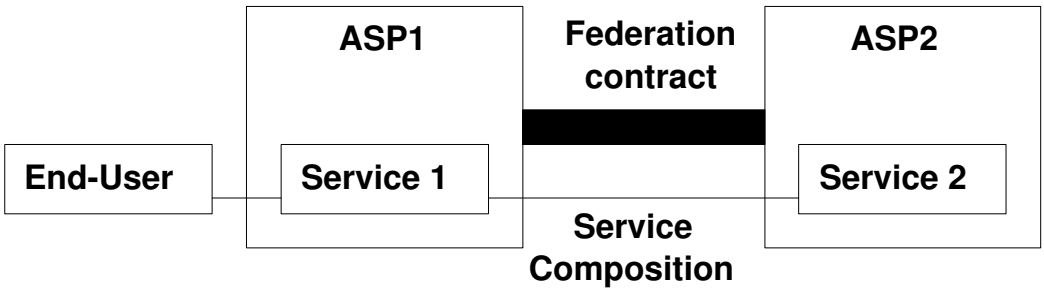

Figure 2.5: Service Composition
[15]

2.4.1 Trading Service

A trader in a distributed environment provides a "yellow pages" service that dynamically maps required service characteristics onto a service interface identifier [17]. A *trading service* implies the use of a trader. The CORBA trading service is one of the services defined by the Object Management Group (OMG); it is an integral part of the CORBA specification.

The CORBA trading service is a commonly used trading service. The CORBA trading service is similar to the well known yellow pages where names are looked for, based on the services they offer. The CORBA trading service is a trading object service that allows an object to be registered with a description of its functionality [18]. This trading service greatly increases the scalability of distributed systems by making services easier to locate.

A service is located by specifying its service properties. The service types are stored in a Service type repository. A service type is a description of a service and it is made up of:

- Type name, which uniquely identifies the service type e.g. email;
- Interface type, which defines the Interface Definition Language (IDL) interface to which an advertised object of this type must conform;
- A collection of property types. It defines attributes of the service offer. A service type can have zero or more property types. A property type consists of a name, type and mode.

The trading service works thus:

1. When a server wishes to advertise its service, it registers a service offer with a trader. The process of advertising a service is called *exporting*.
2. When a client requires a service, it issues a service request to a trader to find a suitable service. A service request contains the service type, properties, and service offer properties. *Importing* is the process of requesting information on a required service. The trader searches its service type repository to match the service request. A list of matched service offers is returned to the client. The list of services contains the service names and object references.
3. After a suitable service is selected by the client, the client can use the service interface identifier to access the server in order to invoke the required service.

Traders can also be linked with other traders to share service offers in their respective databases [17]. This is known as trader federation.

2.4.2 Trader Federation - Database Federation

Trader federation is the interworking of traders in different service provider domain based on an agreement between the service providers. Trader federation is shown in Figure 2.6.

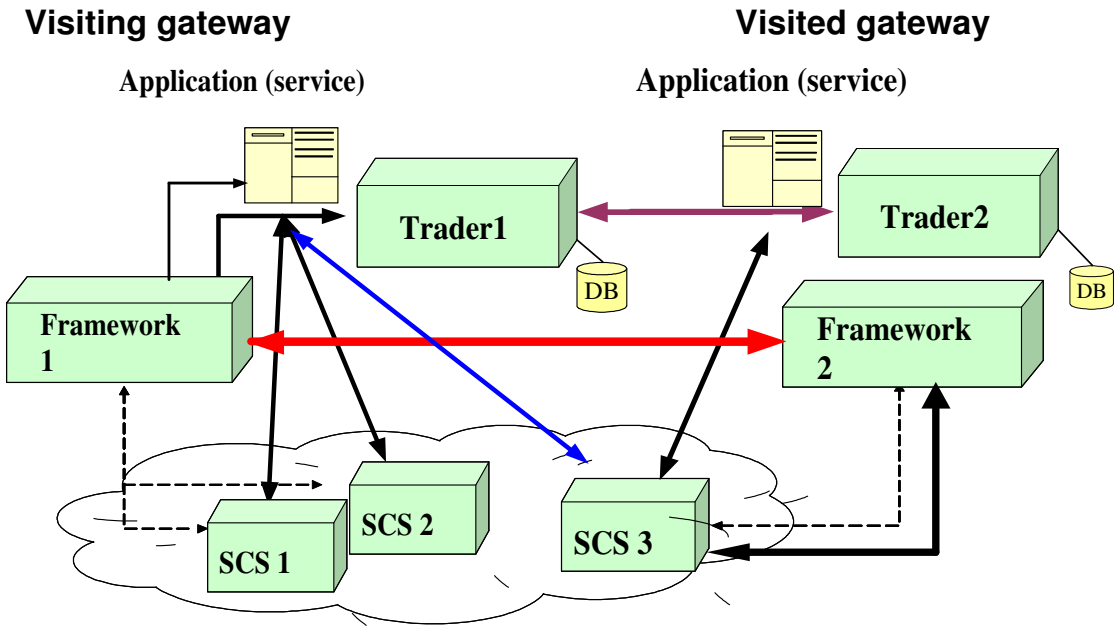

Figure 2.6: Trader Federation

Trader federation allows traders to federate thus allowing them to share service and user profiles. Trader federation enhances the individual effectiveness of the trading service without loss of autonomy.

A trader can make a request to another trader it has a federation with to search out a service that is not available in its own database. In trader federation, each trader has its own database, which contains user profiles and service profiles unique to a service provider's domain.

1. The visiting framework makes a request to its trader to search for a service specified by the application.
2. The trader searches its database and if the service is not found, the trader sends a search request to other traders it has a federation with to search for the service.
3. The matched services and framework references are sent back to the trader and then from the trader to the framework. The application can then use the service via the SCFs.

It is required for federated traders to have access to each other's databases. This approach is referred to as database federation [19]. By having access to each other's databases, the providers can verify the usage policies of a user in another service provider's domain.

Database federation provides the federation of distributed object oriented databases that contain user profiles and service profiles. The databases contain the user / service profiles. One important aspect of database federation is the ability of each database to understand requests from other databases. This concept is discussed in [19].

A good scenario for the use of database federation is that of a user in a service provider's domain wishing to access a service in another service provider's domain. It is assumed that the service providers have a federation contract that enables them to provide each other's subscribers with their services. The visited service provider may want to check if the user is authorised to access the service and the level of QoS at which the service should be offered to the end-user. The visited service provider can therefore access the user's profile to carry out this check. The query for database federation contains the subscriber's name, userid and service name [19].

For the purpose of this research, we assume that trader federation is only possible if there is a federation between frameworks.

2.5 Chapter Summary

This chapter reviewed the concept of service federation and service composition. The different existing implementation approaches to service federation was discussed. We proposed a new approach to framework federation, which can be implemented via the introduction of new framework interfaces. The use of trader federation and hence, database federation for service composition was also discussed. We concluded that framework federation and trader federation are implementation solutions to service federation and service composition in the OSA / Parlay environment.

Chapter 3

Enterprise Viewpoint

The Reference Model for Distributed Processing (RM-ODP) provides a framework that helps to create an architecture within which support of distribution, interworking and portability can be integrated.

The *RM-ODP Enterprise Viewpoint* deals with the requirements of a system. The enterprise viewpoint is used in this chapter to organise the requirements and structure of the system. The enterprise viewpoint requirements are defined in terms of purpose, scope and policies. This enterprise viewpoint introduces the concepts and terminologies necessary to represent the behaviour expected of the teleworking service by other entities within the system. Social and organisational policies are defined in terms of objects, communities and roles of the objects. A *community* is the grouping of objects intended to achieve some purpose.

In summary, the enterprise viewpoint focuses on:

- the purpose / objective of a system;
- the scope of the system;
- the enterprise objects that make up the system, both active and passive objects;
- policies that govern the interactions of the enterprise objects;
- contracts upon which the usage of the system is based.

The enterprise viewpoint of the teleworking service is discussed based on the above mentioned focuses of the RM-ODP enterprise viewpoint.

3.1 Purpose of the Teleworking Service

The objectives of the teleworking system are to have a teleworking service that conforms to the characteristics of the NGN, i.e supports data, voice and video services on one network platform; and make it available for subscription by corporate bodies and individuals. This teleworking service tries to improve the existing teleworking services in terms of:

- **Service federation**: The existing teleworking service offers interaction across service providers' domains but not with high enough security. Interaction between users in different service providers' domains is to be achieved, that is service federation. Service federation will be achieved across providers' domains based on an agreement between the domains as to service and resources usage. The interworking of providers in these domains will be in a secure manner.
- **Service Composition**: The existing teleworking service consists of services such as emails, conferencing, and telephony but in isolated forms. This teleworking service is to be designed as an integrated service that has the simpler services in an integrated form. It will be possible to reuse one or more services to form a compound service and not used in isolation. The teleworking service must consist of existing services like email, video conferencing, shared whiteboard and database access.
- **QoS**: There are existing forms of teleworking service available over the Internet but with no guaranteed QoS. In our design of the teleworking service, the end-user will be able to specify its QoS for each service at subscription phase. The application should be able to decide on whether the specified QoS for communication can take place between two terminals or not based on received terminal information.

This teleworking service is designed in such a way that the user does not need to worry about the configuring of the service especially in the case of a system failure. The service provider deals with service development, repairs and maintenance. The teleworking service will be a user-friendly service. The service logic of the teleworking service is transparent to the end-user.

Enterprise objects, role players in the teleworking service, need to interact with each other to achieve the objectives of the teleworking service. The enterprise objects of the teleworking service are identified to provide a view of service requirements.

3.2 Enterprise Objects of the Teleworking Service

The teleworking service is made up of both passive and active objects. The active objects are the human enterprise objects while the passive objects are the information services such as the database, and the subscription service. Figure 3.1 shows the different enterprise objects of the teleworking service.

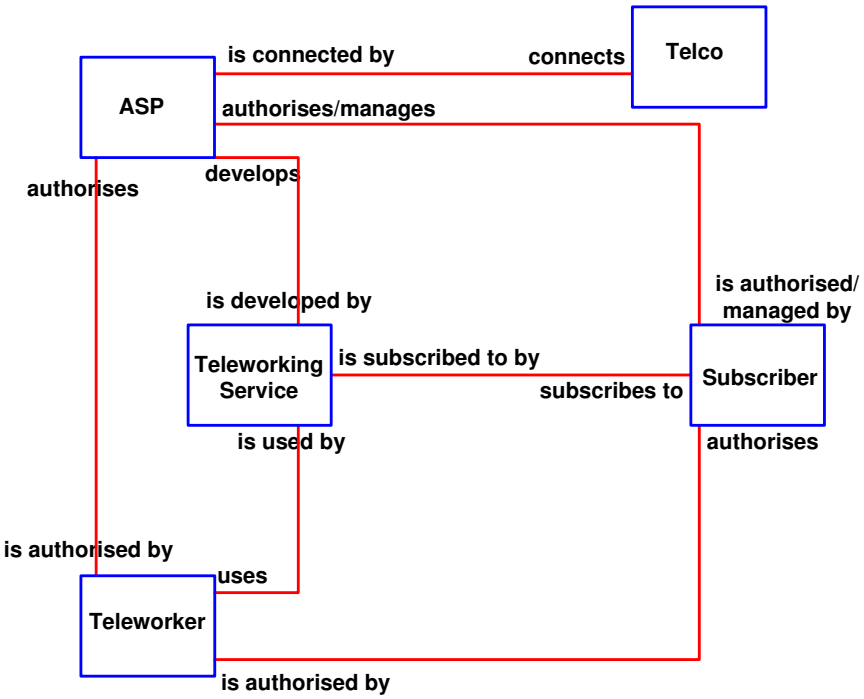

Figure 3.1: Teleworking Service Enterprise Objects and Roles

1. **Telco**

   The *telco* is an enterprise object that provides the needed network connectivity for the teleworking service. The telco provides network connectivity to the developer of the teleworking service, i.e. the trusted third-party. The telco provides all required network resources. The telco acts in the role of the network connectivity provider. The telco bills for network connectivity provision. Routing must be carried out based on a contract between both parties. The telco can develop the teleworking service or make its network open for the use of the teleworking service developer.

2. **Application Service Provider (ASP)**

   The Application Service Provider (ASP) develops and maintains the teleworking service. The ASP makes the teleworking service available to any end-user who subscribes to the teleworking service. The ASP is the ”middle man” between the telco

and the end-user. The ASP manages the consumer in terms of user profile management, security issues (authentication and authorisation) and billing.

3. **Subscriber**

   The *subscriber* is the corporate body that subscribes to the teleworking service. The subscriber is one of the end-users of the teleworking service. The ASP authorises the subscriber to use the service. The direct user of the teleworking service is in turn authorised to use the teleworking service by the subscriber.

4. **Teleworker**

   The *teleworker* is the direct user of the teleworking service. The teleworker is either an individual user of the teleworking service or a user who is registered with the subscriber. As a registered user with the subscriber, the teleworker receives authorisation from the subscriber to use the teleworking service. The teleworker can be a single user or a group of users in the subscriber's domain. In the case of a group of users, the teleworkers are assigned to a Service Assignment Group (SAG). The SAG consists of users with a commonality e.g. geographical location and job type.

5. **Teleworking Service**

   The *teleworking service* is the actual service used by the identified enterprise objects. The teleworking service consists of a subscription service, a billing service and database in addition to being composed of smaller services like telephony and conferencing.

   - **Subscription Service**: The subscription service makes up part of the teleworking service and it takes care of subscriptions to the teleworking service. The subscription manager is notified of any changes made to a user's subscription profile.
   - **Billing Service**: The billing service handles the billing of the teleworker and the subscriber.

Figure 3.2 shows the enterprise objects of the teleworking service and their respective domains. The consumer domain as seen from Figure 3.2 is made up of the subscriber and teleworkers because they are the consumers of the teleworking service. The ASP is in its domain and the network connectivity provider resides in the telco domain. In situations when two or more groups of enterprise objects in different domains, co-operate to meet an objective, they form a domain called a *federation*. An example is that of the ASP and subscriber engaging in providing the teleworking service for the use of the teleworker. The telco and the ASP domains can also be federated as a result of the fact that both objects can develop the teleworking service.

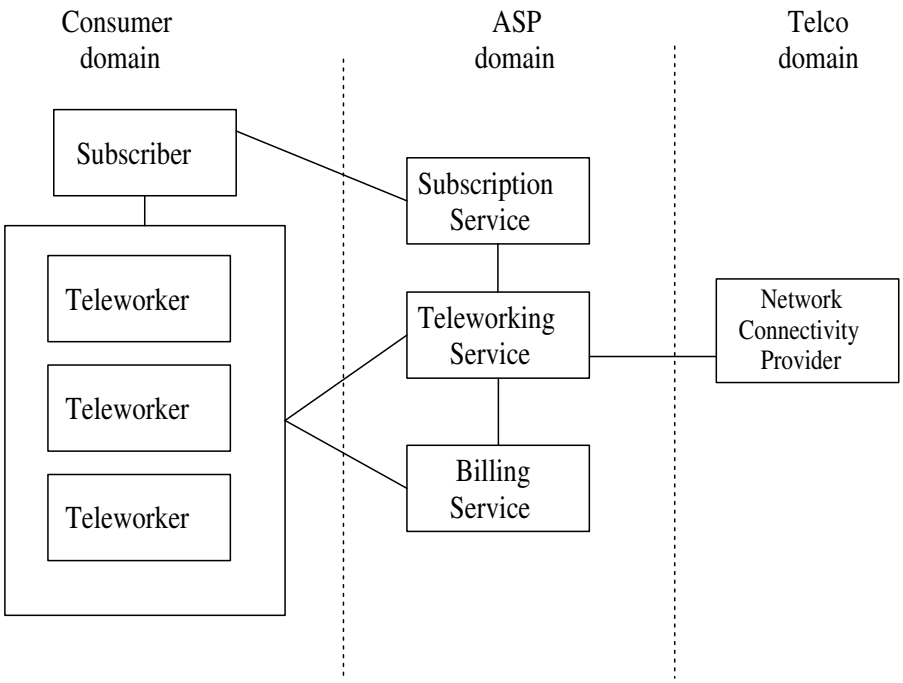

Figure 3.2: Teleworking Service Enterprise Objects and their Domains

3.3 Use Cases

A *use case* represents a functional requirement, showing what happens in a system, but not how it is achieved by the system. The use cases for the working of the teleworking service are shown in Figure 3.3. The three basic active enterprise objects are shown in the use case diagram. The use case diagram shows the roles played by each enterprise object.

3.3.1 ASP Use Cases

1. **Contract service**: The teleworking service is made available to the subscriber or teleworker based on a contract. Both parties involved agree on a contract for service provisioning.

2. **Authorise**: The ASP gives authorisation for service usage to the subscriber and teleworker.

3. **Manage end-user**: The managing of the end-user includes user profile / subscription management and billing.

4. **Coordinate service sessions**: The ASP is in charge of service sessions that the teleworker is in. The ASP starts the service session and also has the capability to suspend, resume or end the service session. Service federation and service composition

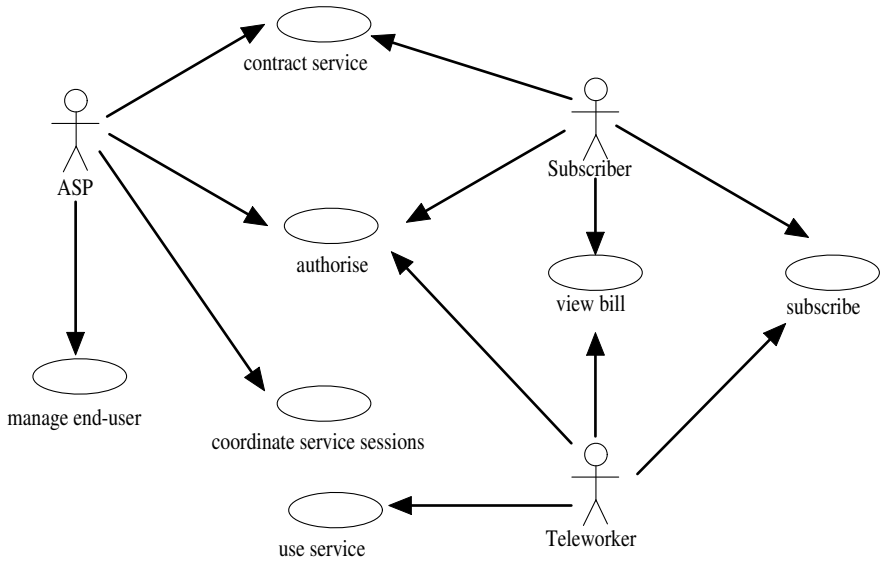

Figure 3.3: Use Case View of the Enterprise Object Roles

are parts of the coordination of service sessions.

3.3.2 Subscriber Use Cases

1. **Contract service**: A contract for service usage exists between the ASP and the subscriber and also between the subscriber and the teleworker.
2. **Subscribe**: The subscriber subscribes to the teleworking service based on an offline contract.
3. **Authorise**: The subscriber receives authorisation from the ASP and gives authorisation to the teleworker for service usage.
4. **View bill**: The subscriber can view billing information.

3.3.3 Teleworker Use Cases

1. **Subscribe**: As an individual user of the teleworking service, the teleworker needs to subscribe to the teleworking service directly with the ASP.
2. **Authorise**: Depending on whether the teleworker is an individual or registered with a subscriber or ASP, authorisation is received directly from either parties before service usage.
3. **Use service**: The teleworker uses the teleworking service based on defined policies.

4. **View bill**: The individual teleworker can view its bill as stated in the offline agreement. The subscriber can decide whether or not to permit the teleworker to view billing information if the teleworker receives authorisation from the subscriber.

3.4 Teleworking Service Usage

The use of the Teleworking service can be divided into three parts, namely:

- Service subscription
- Service usage
- Billing

For the purpose of this research, only the service usage aspect will be dealt with.

3.4.1 How the Teleworking Service Works

Precondition

1. An offline contract negotiation has taken place between the ASP and the subscriber or the teleworker as the case may be. The service contract describes the terms of the provision of the teleworking service to the subscriber. The offline negotiation includes date of start of usage, duration of usage, mode of billing and billing rate. The offline contract negotiation will form the foundation of the online subscription to the teleworking service.

Basic Flow

1. The subscriber or teleworker logs into the ASP's network domain to carry out an online subscription. Once the online subscription has been finalised, the subscriber or teleworker is authorised to use the teleworking service.
2. The subscriber authorises the teleworker to use the teleworking service in the case of the teleworker being registered with the subscriber. Otherwise, the teleworker receives authorisation directly from the ASP.

3. The teleworker goes through almost the same online subscription, as the subscriber did. The difference is that the teleworker does not choose from any list of services, choice of services is determined by the subscriber. The teleworker can only customise the services.

4. The teleworker registers all the terminals that will be needed to access the teleworking service. These terminals must correspond to that specified by the subscriber in the service contract.

5. The teleworker is first authenticated and authorised before each use of the teleworking service. When the teleworker wishes to use the teleworking service, the subscription profile is checked to make sure that the teleworker has authorisation to use the service and also on the specified terminal configuration. The teleworker decides on what terminals to use for different reasons, for example in the case of not being at a terminal when an invitation notification is sent.

6. Assuming a teleworker wishes to invite another teleworker in a different ASP domain to a service session, the teleworker provides the visitor's ASP reference to its ASP who contacts the required ASP.

7. An event notification is sent to the teleworker via the ASP anytime an invitation request to a service session or other event notifications are received.

8. The ASP bills the subscriber for services used by the teleworker.

Multiple users may be located in the same administrative domain; in this case there will be a unique user id for users within this common domain. They may all have the same or different passwords. Multiple users may be grouped based on geographical location and type of job. The interactions of the enterprise objects are governed by policies.

3.4.2 Policies Governing the Teleworking Service

A *policy* is a set of rules related to a particular purpose. Policies govern the interaction of the enterprise objects with each other and with the teleworking service. The policies are defined in terms of obligation, permission and prohibition.

- **Obligation**: An obligation is what the object must do. An obligation is the basis on which communities grant permissions and impose prohibitions.

- **Permission**: A permission is what the object can do. This means that there is no obligation for the behaviour not to occur.

- **Prohibition**: A prohibition is what the object must not do. A prohibition is a prescription that a particular behaviour must not occur.

1. **Telco**

   Obligations

   - The telco must make network resources available to the ASP.
   - The telco must provide adequate and reliable network connectivity to the ASP.
   - The telco must open its network for the ASP to use.

   Permissions

   - The telco can develop the teleworking service.

2. **ASP**

   Obligations

   - The ASP must develop and make the teleworking service available for subscription.
   - The ASP must make the teleworking service available to the subscriber and teleworker in accordance with the agreed-upon service contract.
   - The ASP must bill the subscriber / teleworker for service usage.
   - The ASP must notify the subscription manager of any changes made to a service subscription.
   - The ASP must manage the service sessions.
   - The ASP must support service federation and composition.

   Permissions

   - The ASP can customise the teleworking service (service differentiation).
   - The ASP can interact with other ASPs for service federation purposes.
   - The ASP can create, modify, and delete a subscriber profile upon request from the subscriber.
   - The ASP can negotiate with the teleworker when the specified QoS cannot be met.

   Prohibitions

   - The ASP must not use the network resources for connectivity without authorisation from the telco.

- The ASP must not delete a subscription without notifying the subscriber.

3. **Subscriber**

   Obligations

   - The subscriber must adhere to the service contract for service usage.
   - The subscriber must authorise the teleworker to use the teleworking service.

   Permissions

   - The subscriber can modify and delete its service subscription profile.
   - The subscriber can customise the teleworking service.
   - The subscriber can be subscribed to more than one ASP.
   - The subscriber can add new users to or delete users from the SAG.

   Prohibition

   - The subscriber must not delete its subscription without notifying the ASP.

4. **Teleworker**

   Obligations

   - The teleworker must receive authentication and authorisation from the ASP or the subscriber before using the teleworking service.
   - The teleworker must be authenticated before each service usage.

   Permissions

   - The teleworker can customise the service usage.
   - The teleworker can invite other teleworkers to the same service session.
   - The teleworker can decide on what terminals to use for event notifications.
   - The teleworker can use more than one service within the same service session.

   Prohibitions

   - The teleworker must not delete itself from the service assignment group.
   - The teleworker must not delete its subscription without notifying the ASP.

The above policies are put together in the form of contracts, which are agreed upon by the enterprise objects involved. Contracts exist between two or more enterprise objects and these contracts govern the use of the service provided by either of the two objects. A contract specifies an agreement governing part of the collective behaviour of a set of objects. For example, a federation contract will govern the use of services or interaction across different ASP domains.

3.5 Other Requirements of the Teleworking Service

Apart from the policies, which determine the requirements of the teleworking service, there are other things to be considered in the working of the teleworking service.

3.5.1 QoS Specification

The end-user can specify the QoS for each service or for the whole subscription. The ASP can offer different levels of subscription or service e.g. gold, silver or bronze. Each subscription or service category has a certain QoS associated with it. The QoS can also be specified at the start of each service. In this case, the QoS must be specified at the service level in terms, which are meaningful to the user. For instance, a voice channel can be rated on a scale using four descriptors: CD quality, FM quality, telephone quality and cellphone quality. Other examples are:

- **Email**: The time lapse between sending and reception of an email e.g. should be at most 10 seconds.
- **Audio conference**: Voice quality and time lapse between sending and reception i.e. no perceptible voice delay.
- **Videoconference**: Voice and sound quality

A Sevice Level Agreement (SLA) ensures that the ASP provides the QoS specified by the teleworker. An SLA is a contract between the Service provider and the end-user to create a common understanding about the number of service aspects such as quality levels, support options and a guaranteed level of system performance [20]. The teleworker has no contractual agreement with the telco as to signalling and control QoS specification. The ASP liases with the telco to realise QoS specified by the teleworker. The teleworker is billed with respect to the QoS of each service. A negotiation is carried out between the teleworker and the ASP if the required level of QoS cannot be reached.

3.5.2 Security

Security can be achieved via the OSA / Parlay framework. Authentication and authorisation at the API level ensures that good security is achieved. The ASP is also assumed to be a trusted third party service provider. The choice of architecture also determines the security level of the teleworking service.

3.6 Chapter Summary

We looked at the teleworking service according to the definition of the RM-ODP enterprise viewpoint. We identified the teleworking service enterprise objects and the roles they play in making the teleworking service work right. The interactions of the active enterprise objects with each other and with the teleworking service were discussed. The policies that govern the interactions of the enterprise objects were identified. We looked at other design considerations such as QoS and security. The use cases for the ASP, subscriber and teleworker were shown and expatiated upon. The general working of the teleworking service was discussed. We concluded that for this teleworking service to work right, the general requirements, roles and interactions have to be well defined.

Chapter 4

Information Viewpoint

The RM-ODP information viewpoint describes the information required by an application via the state and structure of an enterprise object [21]. The flow of information within the teleworking service is described in the information viewpoint. The flow of events with respect to the general working of the teleworking service, service federation and service composition are described in this chapter. This teleworking service information viewpoint covers the interactions involved in federation only, it is therefore a partial information viewpoint.

4.1 General Flow of Information

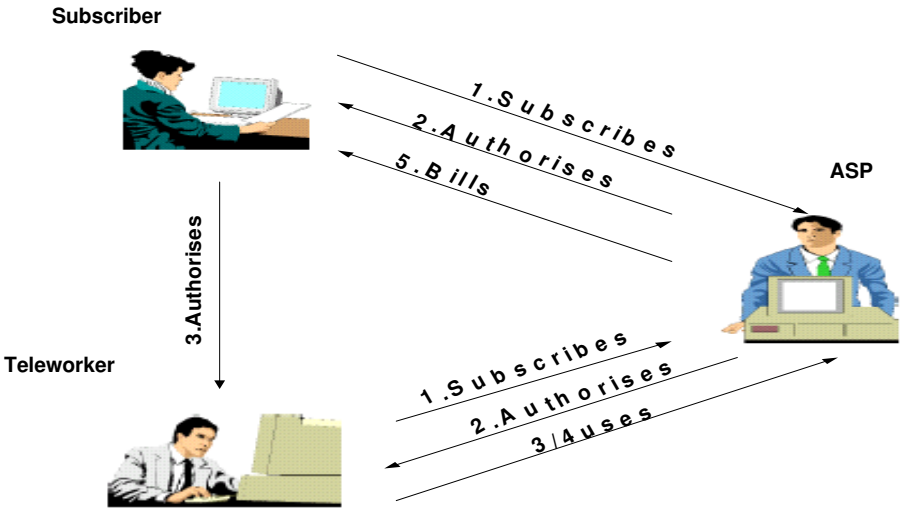

Figure 4.1: Overview of Events in the Teleworking Service [22]

The general flow of information gives a generic view of the working of the teleworking service. This general view is shown in Figure 4.1. There are two scenarios for using the teleworking service. Figure 4.2 shows the sequence diagram view of the first scenario. The second scenario is slightly different from the first scenario as discussed below.

- **Scenario 1**

  A subscriber subscribes to the teleworking service and then employs teleworkers to use the teleworking service.

  1. The subscriber subscribes to the teleworking service with the ASP.
  2. The subscriber receives authorisation to use the teleworking service.
  3. The subscriber authorises the teleworker to use the teleworking service.
  4. The teleworker uses the teleworking service.
  5. The ASP bills the subscriber.

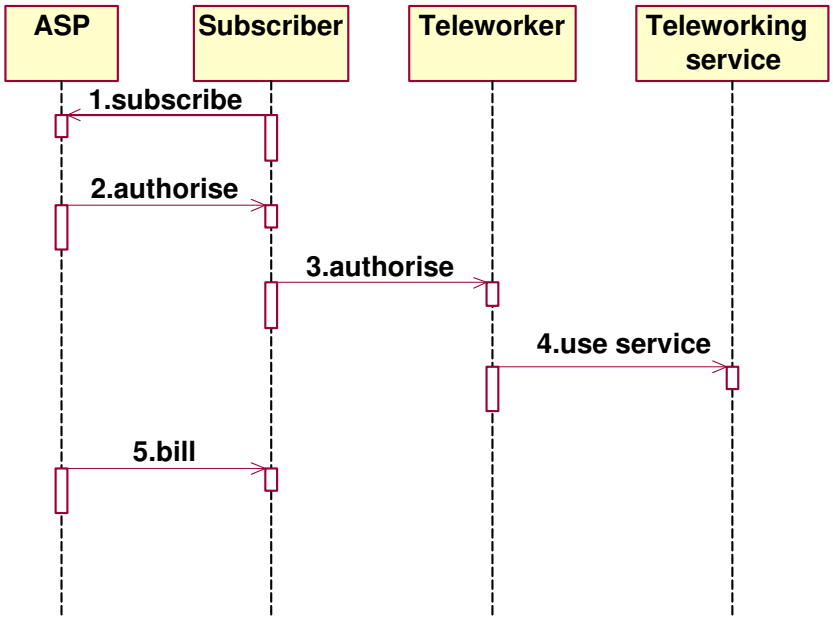

Figure 4.2: Flow of Events for the Working of the Teleworking Service

- **Scenario 2**

  The teleworker is an individual who directly subscribes to and uses the teleworking service.

  1. The teleworker subscribes to the teleworking service
  2. The ASP authorises the teleworker to use the service

3. The teleworker uses the service
4. The ASP bills the teleworker for service usage.

4.1.1 Service Usage

To use the teleworking service, the required service(s) is selected by the teleworker via a Graphical User Interface (GUI). The teleworker has the opportunity to select the QoS level for the service provision at subscription or just before the setup of the selected service. The GUI provides an entry for QoS level for each selected service. The ASP checks for authorisation and then starts the chosen service with the specified QoS. Figure 4.3 shows how a service is started by the teleworker. The server is the subscription server that deals with all user profiles.

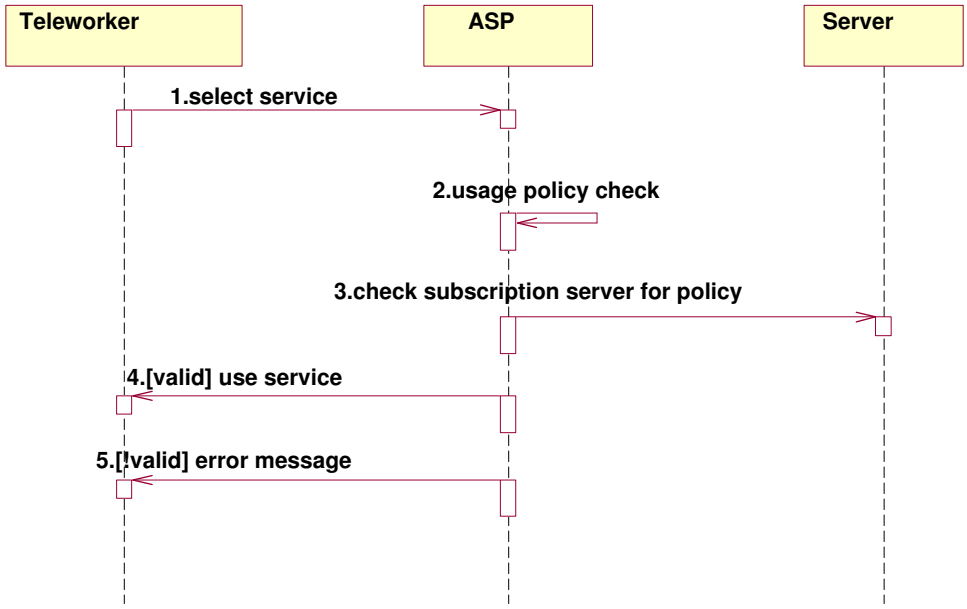

Figure 4.3: Flow of Events for Starting a Service

Precondition

1. The teleworker is a subscribed user of the teleworking service.

Main flow

1. The teleworker selects a service to use and the QoS specifications via a GUI.

2. The ASP carries out checks to see if the teleworker has the permission to use the service and if the teleworker has been authenticated.
3. The ASP checks the teleworker's profile, which is in the subscriber server.
4. If the checks are OK, the ASP allows the teleworker to use the service.

Alternate Flow

1. If the teleworker does not have permission to use the selected service, the ASP sends back an error message to the teleworker.
2. If the teleworker has permission to use the selected service but has not been authenticated, the ASP sends an error message to the teleworker, asking for authentication.
3. If the specified QoS cannot be attained, the ASP sends the teleworker an error message.

4.2 Service Federation via Framework Federation

The flow of information for service federation involves the flow of information for framework federation, which was discussed in Chapter 2. Framework federation requires authentication between frameworks as a first step. The framework represents the ASP and the application plays the role of the teleworker.

4.2.1 Framework Authentication

A framework needs to be authenticated before gaining access to or being able to communicate with other frameworks it has a federation contract with. Framework authentication is described in Figure 4.4. Fw1 stands for framework 1, the visiting framework and Fw2 stands for framework 2, the visited framework.

Precondition

1. A federation agreement exists between both frameworks.

Main Flow

1. Fw1 sends an authentication request to Fw2, requesting to be authenticated.

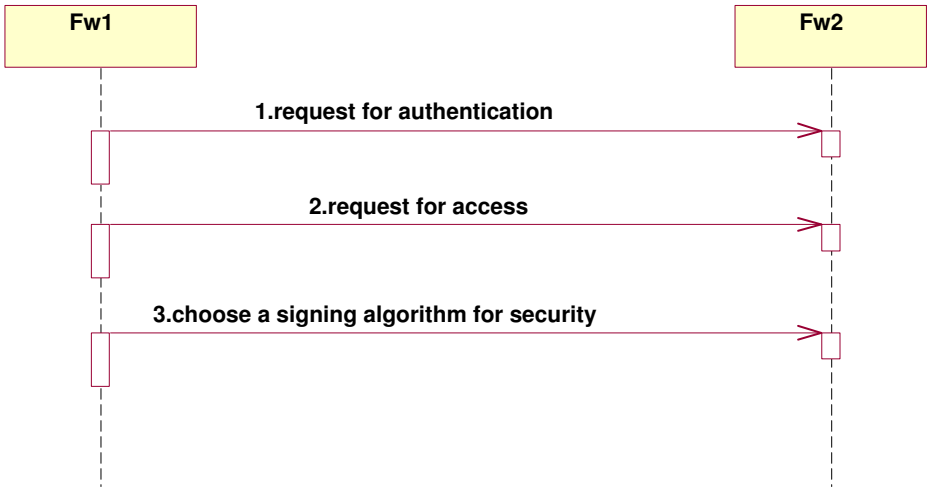

Figure 4.4: Flow of Events for Framework Authentication

2. An authentication version is agreed upon by both frameworks and Fw1 requests for access to Fw2's environment.

3. Fw2 checks if there exists a federation contract between it and Fw1. If there exists a federation contract, a signing algorithm is chosen.

If a federation agreement exists, Fw2 authenticates Fw1 and gives it access permission to its interfaces. After successful authentication of Fw1, both frameworks can communicate with each other. Fw2 does not need to be authenticated immediately after Fw1 has been authenticated unless it wants to pass a request to Fw1.

Alternate Flow

1. If Fw2 does not support the chosen authentication version, an error message is sent to Fw1 with a request for another signing algorithm.

4.2.2 Framework Federation

Figure 4.5 shows the flow of information for framework federation when the App sends an invitation request to its framework.

Precondition

1. A federation agreement exists between both frameworks.

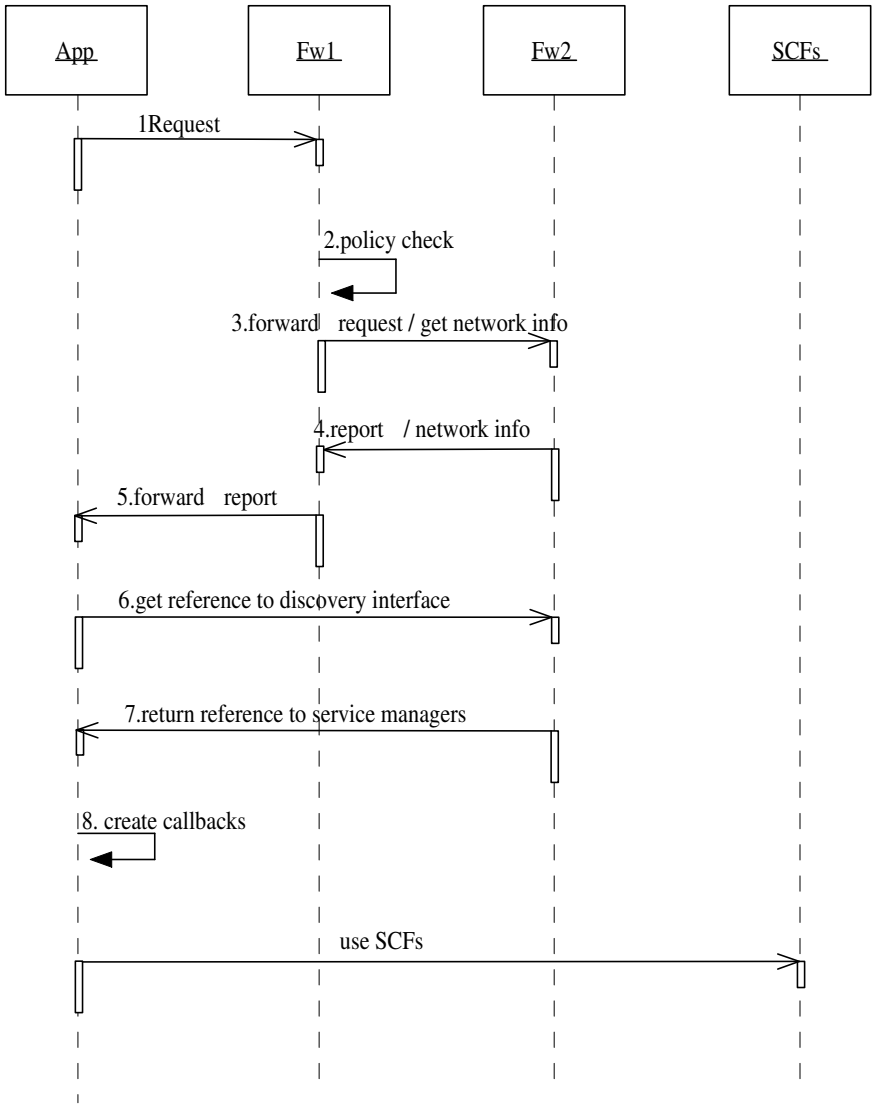

Figure 4.5: Flow of Events for Framework Federation

2. The App has been authenticated by its framework, Fw1.

Main Flow

1. The application, App, sends a request to its framework, Fw1, to use SCFs in Fw2's domain. The request contains information such as the destination address and framework address of the invited teleworker and list of required SCFs.
2. On receiving an invitation request, Fw1 checks if it has a federation agreement with Fw2 and if App is authorised to create a service instance in Fw2's domain.
3. If a federation agreement exists between Fw1 and Fw2, Fw1 forwards the request to Fw2. Fw1 may include a request for QoS supported by Fw2 in the request.

4. Fw2 checks if a federation exists between it and Fw1 and if yes, Fw2 checks to see if Fw1 has been authenticated. There can be provision for Fw2 to check if the App is authorised to create a service manager instance in its domain. If Fw1 has been authenticated, Fw2 returns a report that contains permission for the App to access its interfaces. The report for QoS supported is also returned to Fw1.

5. Fw1 forwards permission report to App.

6. On receiving the permission report, App invokes a request on Fw2 for the reference to its discovery interface to discover the available SCS. The App selects the desired SCFs on getting the discovery interface reference of Fw2.

7. Fw2 creates a service manager for the selected SCFs as required and returns the service manager reference to the App.

8. App creates the callbacks to receive responses and notifications from the SCFs. The SCFs include the mobility, terminal capability and call control SCFs [1].

If the request is an invitation request, on receiving reports from the SCFs, the application can check for terminal compatibility to ensure good QoS communication. Once the application has been authenticated by its framework, it is not necessary for the visited framework to authenticate the application again [2]. The federation contract permits the visiting gateway to carry out authentication on behalf of the visited framework. The visited framework may want to authenticate the application itself, this is specified in the report that contains permission for the application to access Fw2's interfaces.

Alternate Flow

1. If Fw1 has not been authenticated, Fw2 sends an error message to Fw1 and requests for authentication.

2. If the returned reports show that communication cannot occur between both teleworkers due to difference in QoS supports, the application can either cancel the invitation request or a QoS negotiation can be made between both providers to find a reference QoS level for communication via the network information interface.

[1] The sequence diagrams for the specific SCFs are shown in a latter chapter

[2] The sequence diagram for the authentication of the application by the framework is shown in Appendix C

4.3 Service Composition via Trader Federation

Service composition involves trader federation, this is shown in Figure 4.6. Trader1 is in Fw1's domain and trader2 is in Fw2's domain.

Preconditions

1. A federation agreement exists between both frameworks
2. A federation agreement exists between the two traders. The federation agreement between the traders is governed by the agreement between their frameworks.
3. The teleworker is already using a service instance.

Main Flow

1. App sends a request for a new service to Fw1. The request contains the name and properties of the required service.
2. Fw1 checks to see if App is authorised to use the selected service.
3. Fw1 uses its trader service, trader1, to search for the service in its domain.
4. Trader1 searches for the required service in its service repository type (database).
5. Assuming the service does not exist in this trader1's domain, trader1 sends a request to other traders it has a federation with.
6. Trader2 searches its database for the service. Trader1 sends back a list of services that match the specified service property. The list contains the services and their corresponding object references. The object reference is the reference to the ASP offering the service.
7. Fw1 forwards the service list to the application.
8. App selects required service from the list.
9. App sends a request to Fw1 for the reference to the service.
10. Fw1 forwards the reference request to Fw2.
11. Fw1 gets the reference from Fw2
12. Fw1 forwards reference to application.

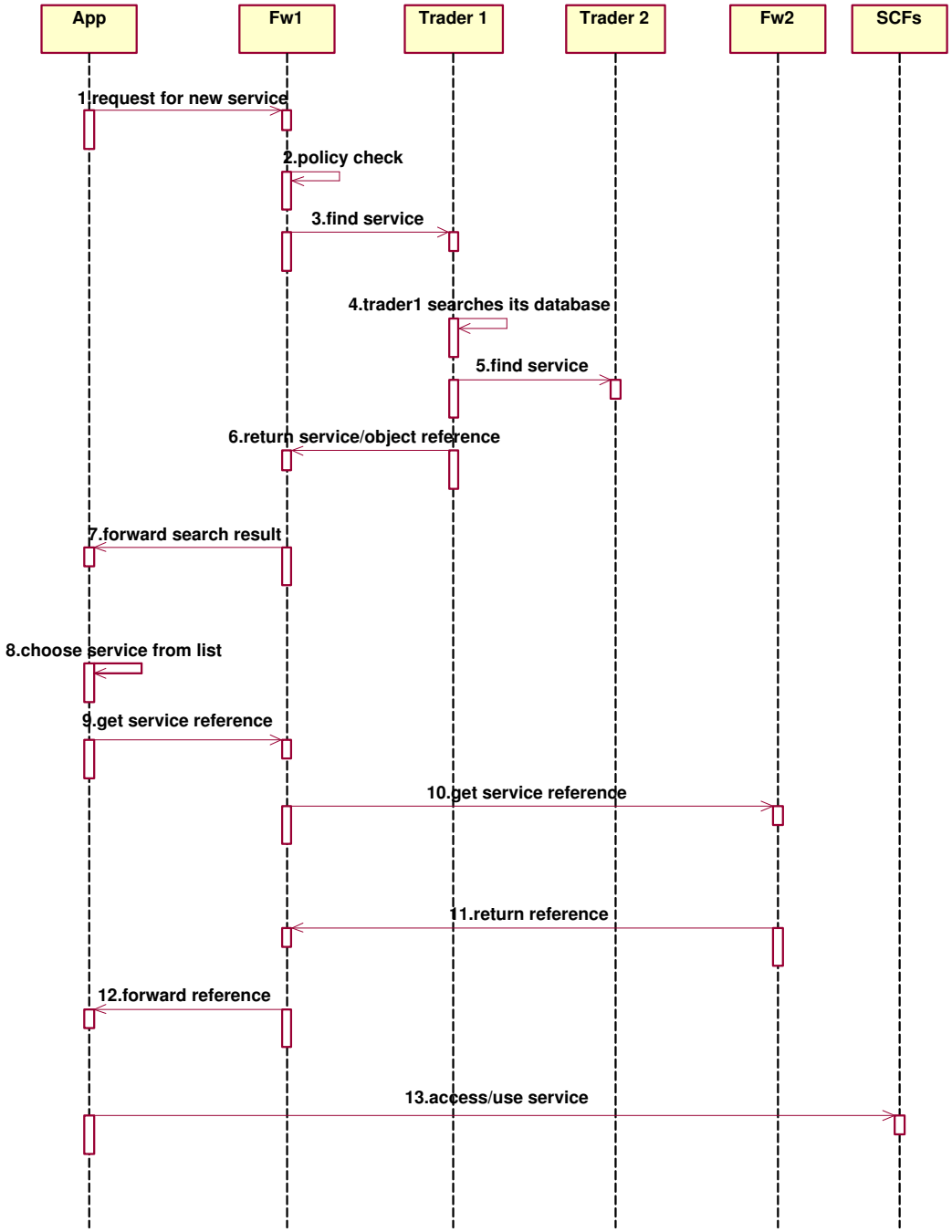

Figure 4.6: Flow of Events for Trader Federation

13. Application now has access to the required service.

Framework federation is required for access to the domain of the framework that offers the service.

4.4 Chapter Summary

This chapter discussed the flow of information in the teleworking service design. A partial information viewpoint that focuses on federation and composition was discussed. The basic operation of the teleworking service in terms of subscription, authorisation, usage and billing was shown. The flow of events from service federation (framework federation) to service composition (trader federation) was described. Framework authentication as a first step to framework federation, service selection and trader federation were discussed via the use of sequence diagrams. Framework federation was seen as a pre-requisite for service composition. Based on a desired architecture, these events can be viewed in terms of computational objects, interfaces and methods.

Chapter 5

Computational Viewpoint

The RM-ODP computational viewpoint is used to specify the functionality of a service in a distribution transparent manner [21]. The computational viewpoint focuses on the distribution of a system through functional decomposition into objects which interact at interfaces. The computational viewpoint is object-based i.e.,

- objects encapsulate data and processing;
- objects offer interfaces for interaction with other objects;
- objects can offer multiple interfaces.

The choice of open network architecture is made in the computational viewpoint. This decision leads to the choice of computational objects, interfaces and the operations that are invoked on these interfaces. The OSA / Parlay architecture is the chosen architecture for the design of this teleworking service.

5.1 New Framework Interfaces

Service federation and service composition in the OSA / Parlay environment are achieved through the interworking of two or more OSA / Parlay frameworks, framework federation. The present OSA / Parlay framework specifications [22] specify:

1. the authentication of applications by the framework and vice versa;
2. the interactions between the framework and the application in terms of authentication and access for usage of SCFs;

3. the interactions between the framework and the SCFs such as service registration.

The OSA / Parlay specifications do not specify:

1. the interactions between two or more frameworks;
2. the interaction between an application and a framework in terms of requests different from authentication and access requests;
3. the authentication operations between frameworks.

With the gaps in the present OSA / Parlay specifications, there is the need to introduce new framework interfaces that support the above mentioned interactions that are not presently supported by the present specifications. Table 5.1 shows the new framework interfaces and their input and output parameters

Table 5.1: New Framework Interfaces

Framework Interfaces	**Interfaces**	**Operations**	**Parameters**
Request Interface	IpRequest	(a)invitationRequest(), (b)useSCF(), (c)newServiceRequest(), (d)getServiceRef()	in:originatingAddress, destinationAddress-Set(terminalAddr,FwAddr,userid) serviceName, serviceProperties, SCFName return:requestid
Network Information	IpNetworkInfo	(a)getNetworkInfo()	in:SCFName(terminal capabilities, user status);return:reports

1. **Request Interface**

   The *IpRequest* interface on the framework supports the passing of requests to the framework by the application. The methods on this interface are: *invitationRequest()*, *newServiceReq()*, *getServiceRef()*, and *useSCF()*. The in parameters for these methods include: originating address and destination address (terminal address, ASP address), SCF list, service name and service properties. The returned parameters include an assignment Id for the request, service list, interface list and gateway reference. The *invitationRequest()* is passed when the teleworker wishes to invite another teleworker to a service session. The App requires the location of the visited terminal for the routing of the call and also other SCFs like the call control SCF. The originating and

terminal addresses have to be included in this request, the SCFName parameter is optional. The *useSCF()* method is used when the application logic wishes to use SCFs from another framework domain and not necessarily to communicate with another terminal in the visited domain. The *newServiceReq()* is used to request for a new service instance and the *getServiceRef()* for reference to the service instance. The IDL definition of this interface is shown in Appendix A[1].

2. **Network Information Interface**

   The *IpNetworkInfo* interface supports the passing of requests between federated frameworks. The *getNetworkInfo()* method has as in parameters; destination terminal address, list of required SCFs (terminal capability, mobility, multiparty, etc). A report that contains permission for the application to access the visited gateway's framework is returned. This interface supports framework to framework requests like QoS negotiation and passing of the application's request.

The other interfaces used for the design of this teleworking service are the interfaces on the terminal capability, mobility, generic call control and conference call control.

5.2 Sequence Diagrams

The sequence diagrams show the information flow between the OSA / Parlay computational objects. The computational objects, interfaces and operations are shown in the sequence diagrams.

5.2.1 Starting a Service

The starting of a service instance is shown in Figure 5.1. The sequence diagram shows how the framework creates the service manager of an SCF, thus allowing the application to access the SCF. This does not require any framework federation as it occurs within the application's provider's domain. The generic call control SCF is used to illustrate the start of a service.

1. App selects a service on Fw1's *IpServiceAgreeementMangement* interface.
2. App invokes the *signServiceAgreement()* operation on the framework's *IpServiceAgreement* interface to seal the service agreement.

[1]The IDL definition for each interface is shown in Appendix A

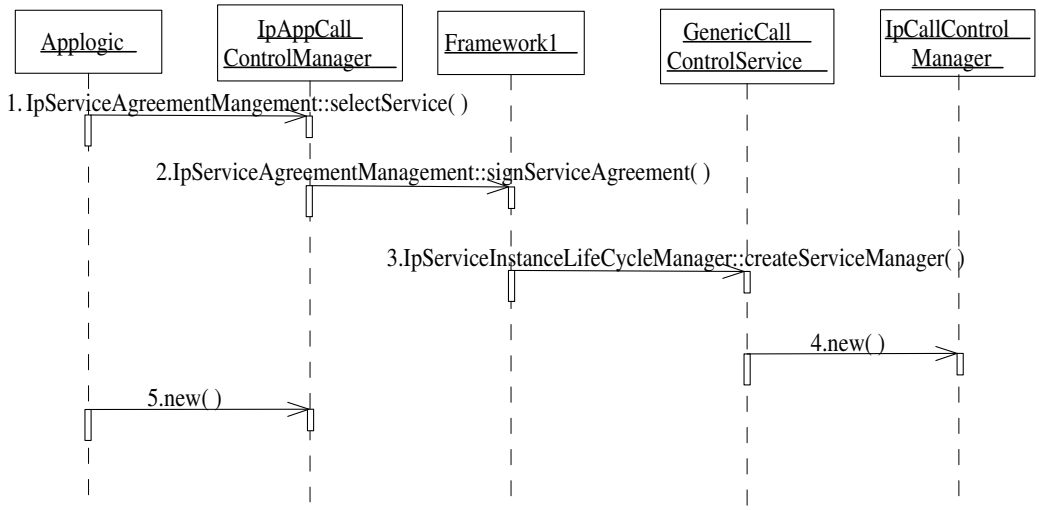

Figure 5.1: Sequence Diagram for Starting a Service

3. The framework invokes the *createServiceManager()* operation to start the selected service.

4. A new service manager is created. Not all SCFs have a separate object as the service manager. An example is the terminal capability SCF. No service manager is created, the App gets access to it via the discovery interface and creates a callback to receive responses from the SCF.

5. App creates the call back object for the service if the SCF has a service manager.

5.2.2 Framework Authentication

Figure 5.2 shows the computational view of how frameworks authenticate each other before interacting, based on the federation contract that exists between them.

1. Fw1 requests to be authenticated based on a chosen authentication version by invoking the *initiateAuthenticationWithVersion()* operation on Fw2.

2. Fw1 uses the *requestAccess()* operation to request for access to interfaces on Fw2.

3. Fw1 and Fw2 agree on a signing algorithm for security purposes and QoS specification.

The framework authentication is carried out to ensure a secure interaction between federated frameworks. Fw2 has to be sure that the framework requesting for authentication has the

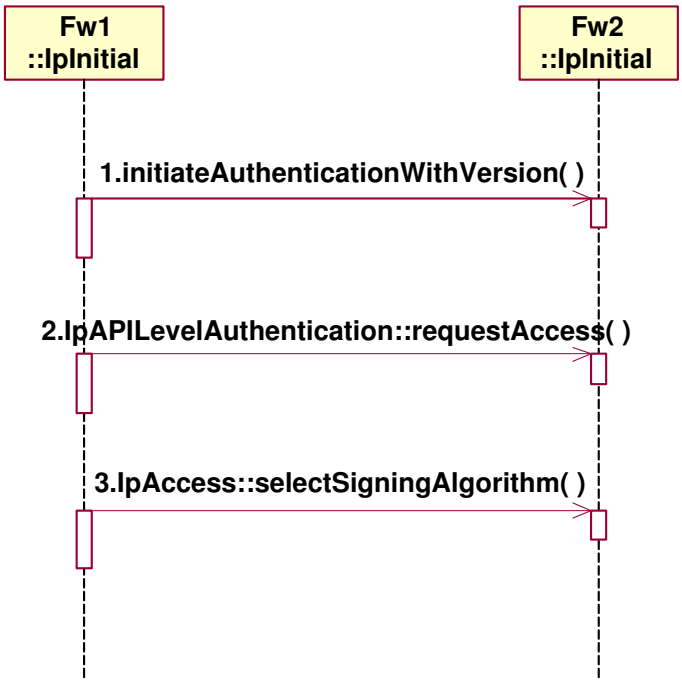

Figure 5.2: Sequence Diagram for Framework Authentication

permission to ask for authentication. The authentication version must be supported by both frameworks.

5.2.3 Framework Federation

The sequence diagram in Figure 5.3 shows the use of framework federation to achieve service federation. The computational objects, interfaces and operations are shown in the figure.

1. App sends a request to the *IpFwRequest* interface on Fw1, to use SCFs in the visited gateway and hence, communicate with a terminal in Fw2's domain. The request contains the ASP (framework) and terminal addresses of the inviting and invited teleworker, the destination address, and the required SCFs.

2. Fw1 requests for the network information of the given destination address by invoking the *getNetworkInfo()* operation on the *IpFwNetworkInfo* interface of Fw2.

3. Fw2 returns a report that contains the permission for the App to access Fw2's interfaces and also supported QoS level.

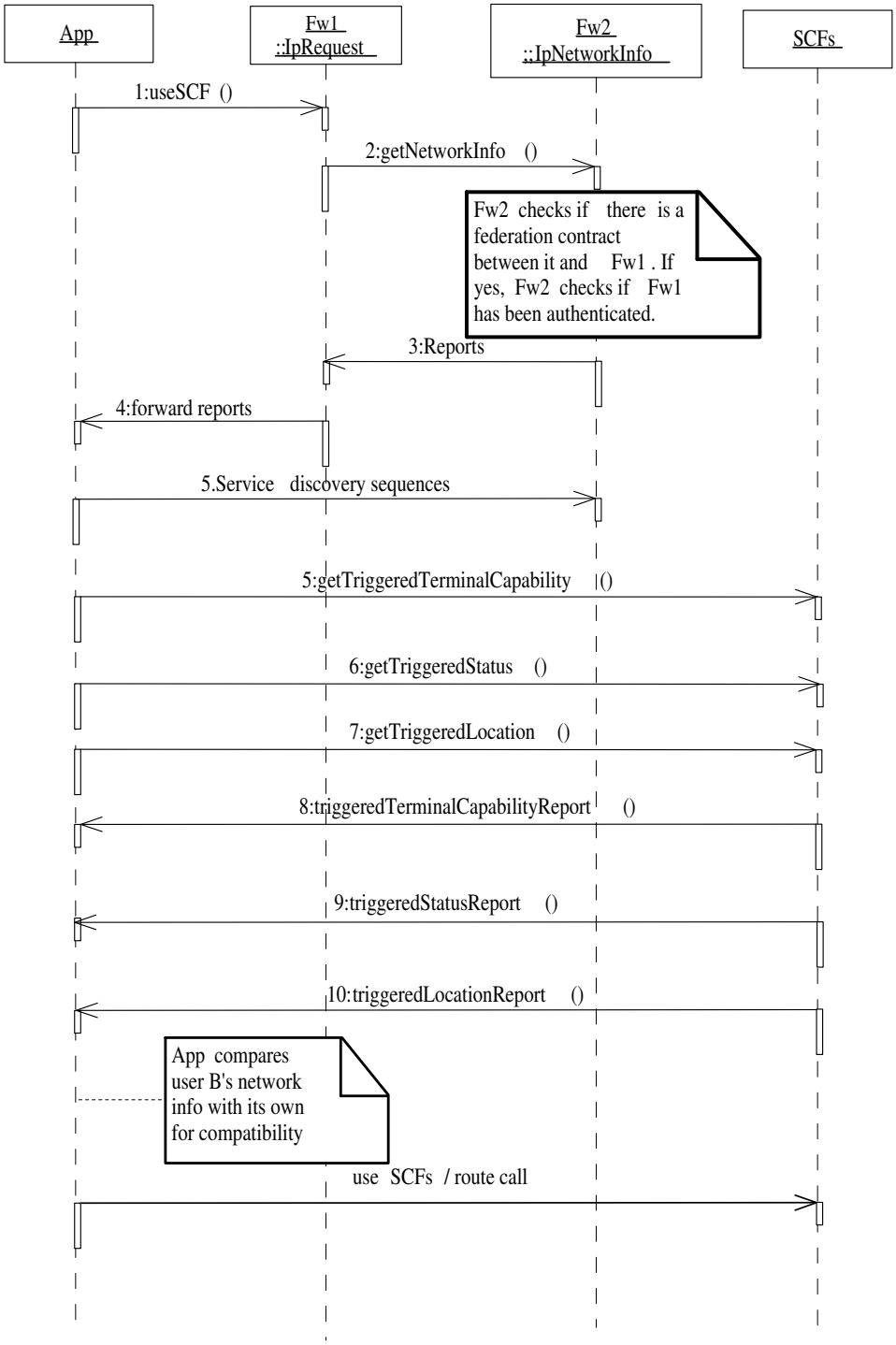

Figure 5.3: Sequence Diagram for Framework Federation

4. Fw2 forwards the report to the App.

5. The sequences for the discovery and accessing of SCS occur next. The sequence diagram in Figure 5.1 gives an instance of how an application uses an SCF [2].

[2]The sequence diagrams for service discovery is shown in Appendix C

6. The App makes requests to the required SCFs. The SCFs include the call control SCF for call routing, the terminal capability SCF for terminal capability of the terminal and mobility SCF for location of the terminal. A report request for triggered terminal capability. This message requests for the updated terminal capability report of the terminal.
7. A request for triggered user status.
8. A request for triggered user location.
9. The SCFs return the reports for each of the requested network information to the App. These reports are returned simultaneously. The report for terminal capability.
10. The report for user status.
11. The report for user location.
12. App has access to the destination terminal and can use the call control SCF to route calls to the destination terminal and use other SCFs from the domain.

For an invitation request, the App compares the returned reports from the SCFs with its own for compatibility to achieve good QoS communication. The App can also pass the *listInterfaces()* operation to Fw2 to have a knowledge of supported interfaces.

5.3 Trader Federation

The sequence diagram for the trader federation is shown in Figure 5.4.

1. App invokes the *newServiceReq()* operation on Fw1, requesting for access to a new service instance. Included in the operation are the properties of the service and service name.
2. Fw1 makes a request to its trading service to look for the service. Trader1 does not find the service in its database
3. Trader1 forwards the request to another trader it has a federation with i.e., trader2. Trader2 searches for the service in its domain and returns a list of matched services to trader1.
4. Trader1 returns a list of matching services to Fw1.
5. Fw1 forwards the search result to App. The result is a list of matching services with respective object references.

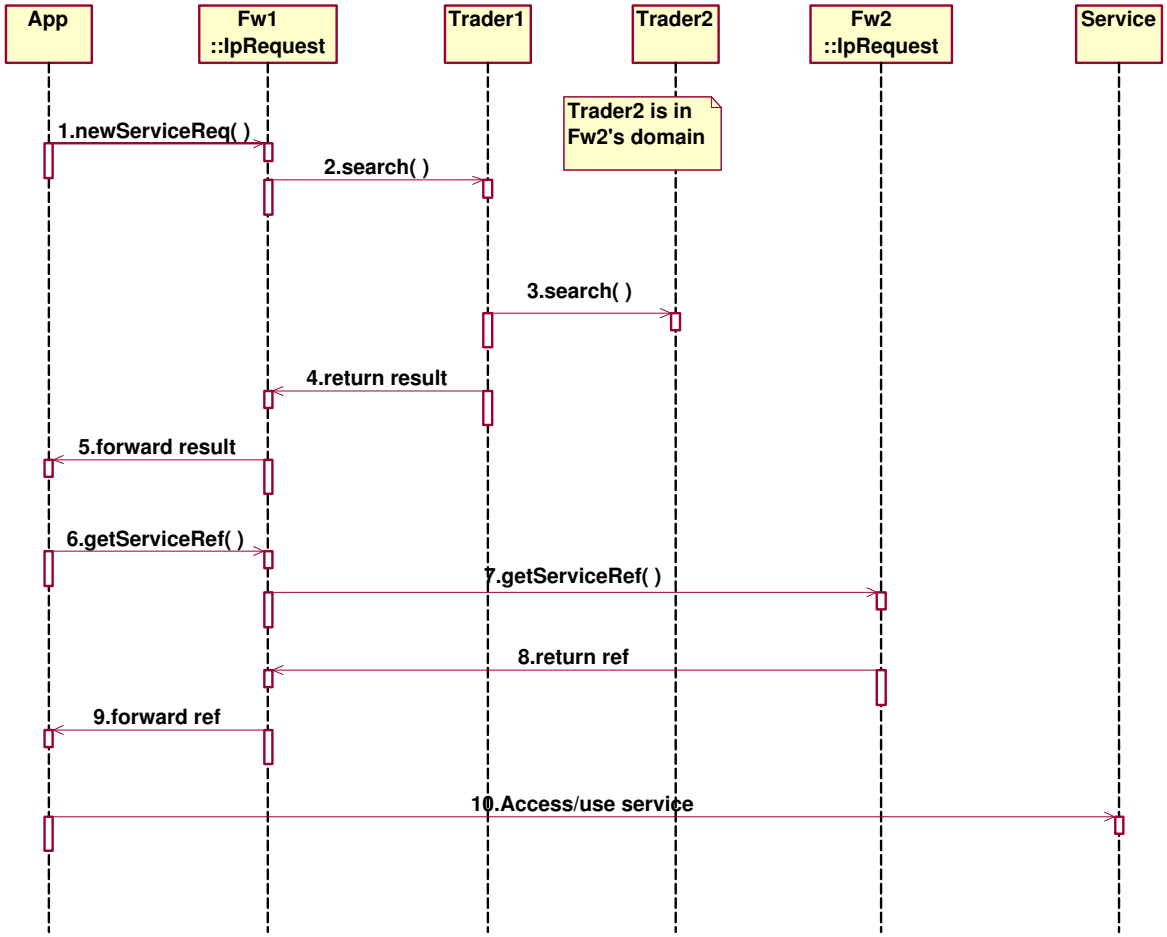

Figure 5.4: Sequence Diagram for Trader Federation

6. App selects a service, which contains the reference to its framework. App invokes the *getServiceRef()* to get the reference to the service interface.
7. Fw1 forwards the request to Fw2.
8. Fw2 returns the reference to the service.
9. Fw1 forwards the reference to App
10. App accesses and uses the service.

5.4 Chapter Summary

This chapter presented the design of the teleworking service in the OSA / Parlay environment. The design of the teleworking service based on how a service is started, framework federation and trader federation was described. The new framework interfaces; IpRequest and IpNetworkInfo were used in the design. The parameters of the new framework interfaces were identified. The new framework interfaces make service federation and composition possible.

Chapter 6

Engineering and Technology Viewpoints

The proof of concept for service federation and service composition are described in this chapter. The proof of concept for this teleworking service was carried out on the SATINA-NGN platform, using the Common Object Request Broker Architecture (CORBA) provided distributed programming environment.

6.1 SATINA Platform

The South African TINA (SATINA) platform is divided into Local Area Networks (LANs); with each of the computers in the LAN acting as the role players in the NGN business model. The physical transport network within the SATINA-NGN platform is composed of different transport technologies such as the IP / MPLS.

6.2 Common Object Request Broker Architecture (CORBA)

The Common Object Request Broker Architecture (CORBA) is an open distributed object computing infrastructure. CORBA enables communication across different computing nodes on the SATINA platform. CORBA supports network programming tasks such as object registration, location, error-handling, and dispatching of operations. CORBA defines a model that allows interoperability between distributed objects on a network, in a transparent, network and technology independent manner.

The CORBA architecture is based on clients requesting the services from distributed objects through a well-defined interface, by issuing requests to the objects in the form of events. CORBA objects are accessed through the use of an IDL. The IDL is used to define

interfaces, their attributes, methods and parameters to the methods within the interface.

The CORBA architecture consists of an ORB, which is responsible for all of the mechanisms required to find the object implementation for the request, to prepare the object implementation to receive the request, and to communicate the data making up the request.

The TAO ACE ORB is used in this research. The TAO trading and naming services are used for searching out services and retrieving object references respectively. C++ is the adopted programming language for the development of the computational objects and interfaces.

6.3 Application Example

Service federation and service composition are illustrated using the audio conferencing and shared whiteboard services. A teleworker decides to use the audio conferencing service together with other teleworkers, who are in the same and different ASP domains. Service federation is used to allow the teleworkers especially within different ASP domains to interact. The teleworker who initiated the audio conference decides that opinions of other teleworkers are needed for the editing of a document. Hence, the teleworker starts up the shared whiteboard service while still using the audio conference service. The start up of the whiteboard service illustrates service composition and the use of service composition with the other teleworkers shows that service federation can be used together with service composition.

6.3.1 Description of the Audio Conference Service

An audio conference service allows two or more people to communicate with each other via voice alone. Audio conferencing is simply a telephone conversation with two or more parties. It is a multiparty service unlike the conventional phone call. Audio conferencing can be used to hold remote departmental meetings, executive board meetings, carry out supervisory work, receive updates from a field sales representative, and so on. All these job activities are part of the job activities that teleworking service can be used for as mentioned in the introduction. This makes audio conference a good constituent of the teleworking service.

The audio conferencing takes place in a conference room. The audio conference illustrated here is a prearranged one, i.e., the participants decide on a date and time in which to have the conference. Each participant is given a password with which to log into the conference

room. An announcement is played via an interactive voice response system each time a new participant enters or leaves the conference room. The ASP of the audio conference initiator controls the audio conference. Figure 6.1 shows how service federation is achieved. The dotted lines show that the teleworkers are in the same conference room.

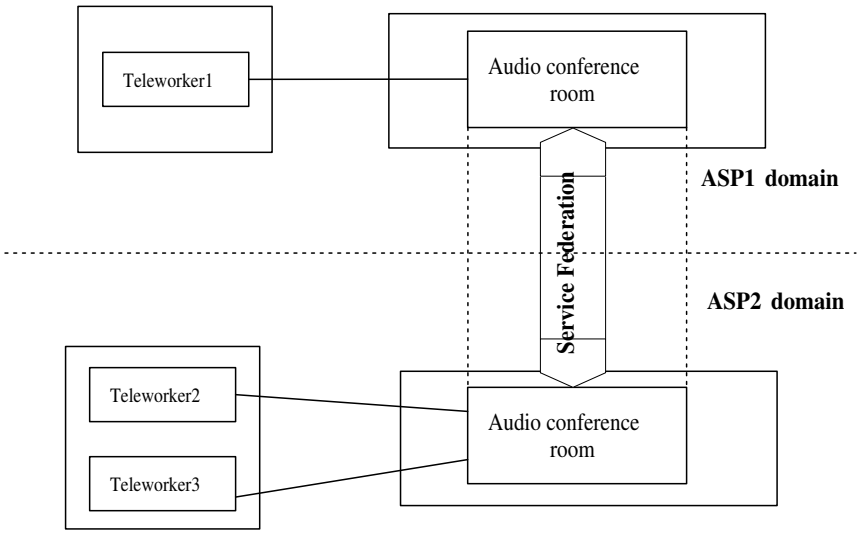

Figure 6.1: Service Federation Using the Audio Conference Service

Teleworker2 is the initiator of the audio conference service. This implies that teleworker3 and teleworker1 are invited to the audio conference by teleworker 2. All three teleworkers are in the same conference room with ASP2 in charge of the service sessions.

6.3.2 Description of the Shared Whiteboard Service

The shared whiteboard service allows users to draw and make annotations on a shared whiteboard. The shared whiteboard user uses an electronic pen to draw on the whiteboard. There is only one pen, thereby controlling the number of people that can draw at a time. The teleworker that needs to edit a document displays the document, which comes to the view of all the participants. Each participant uses the pen to make annotations on the document. The annotations are not saved, only the teleworker with the original document can save changes made to the document. When service composition is achieved, the participating teleworkers can make contributions to the document and hear each other's contributions at the same time. Service composition using the audio conference and shared whiteboard services is shown in Figure 6.2.

For service composition, the shared whiteboard service is within ASP1's domain, the audio conference and whiteboard sharing services are offered by different ASPs.

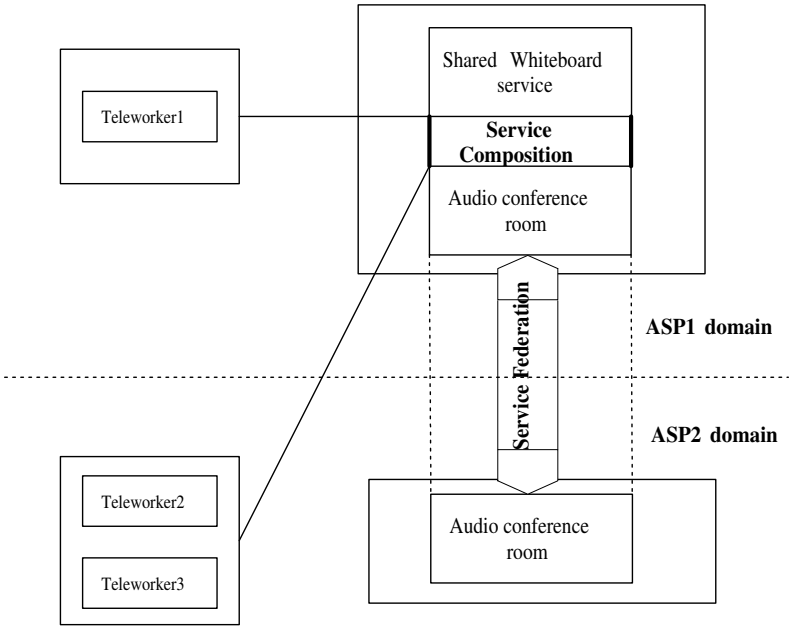

Figure 6.2: Service Composition with Audio Conferencing Service and Shared Whiteboard Service

The characteristics of the required OSA / Parlay SCFs for the audio conference and shared whiteboard services are given in Table 6.1

Table 6.1: Services, Characteristics and Needed SCFs

Applications	**Characteristics**	**SCFs**
Audio Conference	(a) Multiparty Service; (b) Interactive session; (c) Voice channel as media	(a) Conference call control; (b) User Interaction
Shared Whiteboard	(a) Multiparty service; (b) Voice and data media channels	(a) Multiparty; (b) Data session control SCF

6.3.3 Sequence Diagrams for the Used SCFs

The SCFs used for the implementation are those listed in Table 6.1. The terminal capability and mobility SCFs are used for retrieving user network information. Unlike in the case of the call control SCF as shown in Figure 5.1 of chapter 5, the terminal and mobility SCFs do not have a separate object as the service manager. The framework does not create a service manager. Only the application creates a callback object for responses and notifications

- **Terminal Capability SCF**

  Figure 6.3 shows the sequence diagram of how the application gets the terminal capability reports from the terminal capability SCF.

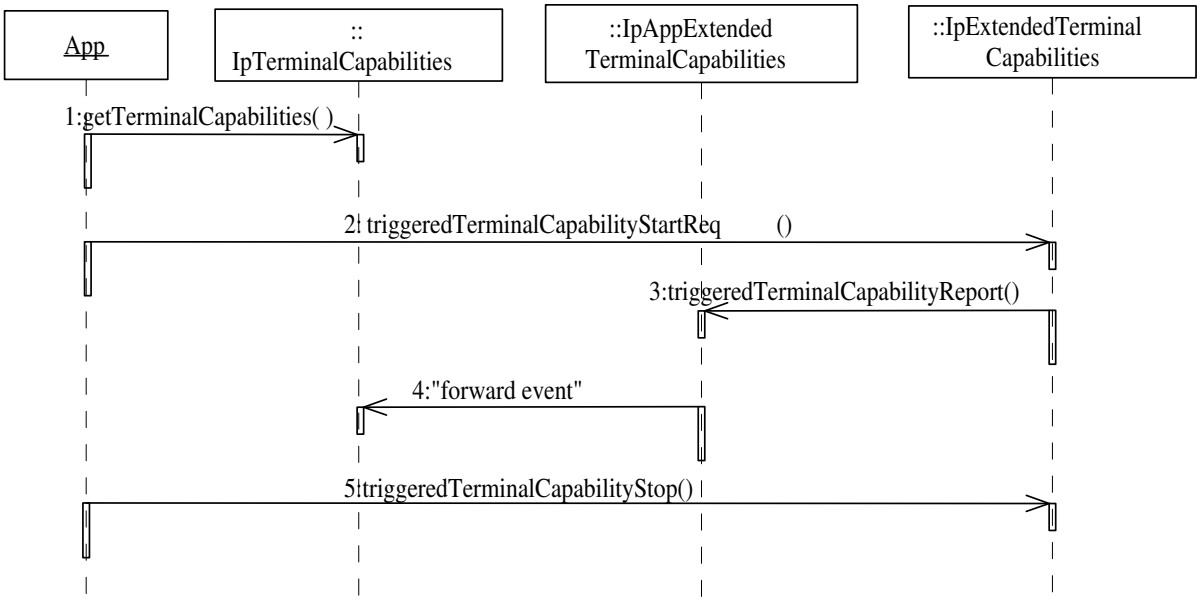

Figure 6.3: Sequence Diagram for Terminal Capability Retrieval

1. The App retrieves the terminal capability of a terminal.
2. The terminal capabilities changes are started to be monitored.
3. The terminal capabilities have changed and they are reported as requested.
4. The report is forwarded to the App.
5. The terminal capability monitoring is stopped.

- **Mobility SCF**

  Figure 6.4 shows how the user status and location of the terminal are retrieved by the application.

1. The App requests for the triggered user status of the destination terminal.
2. This message is used to start triggered status reporting for the destination terminal.
3. This message is used to start triggered location reporting for one or several users.
4. When there is a triggered location change, the *triggeredLocationReport()* method is used to pass the location of the user to its call back object.

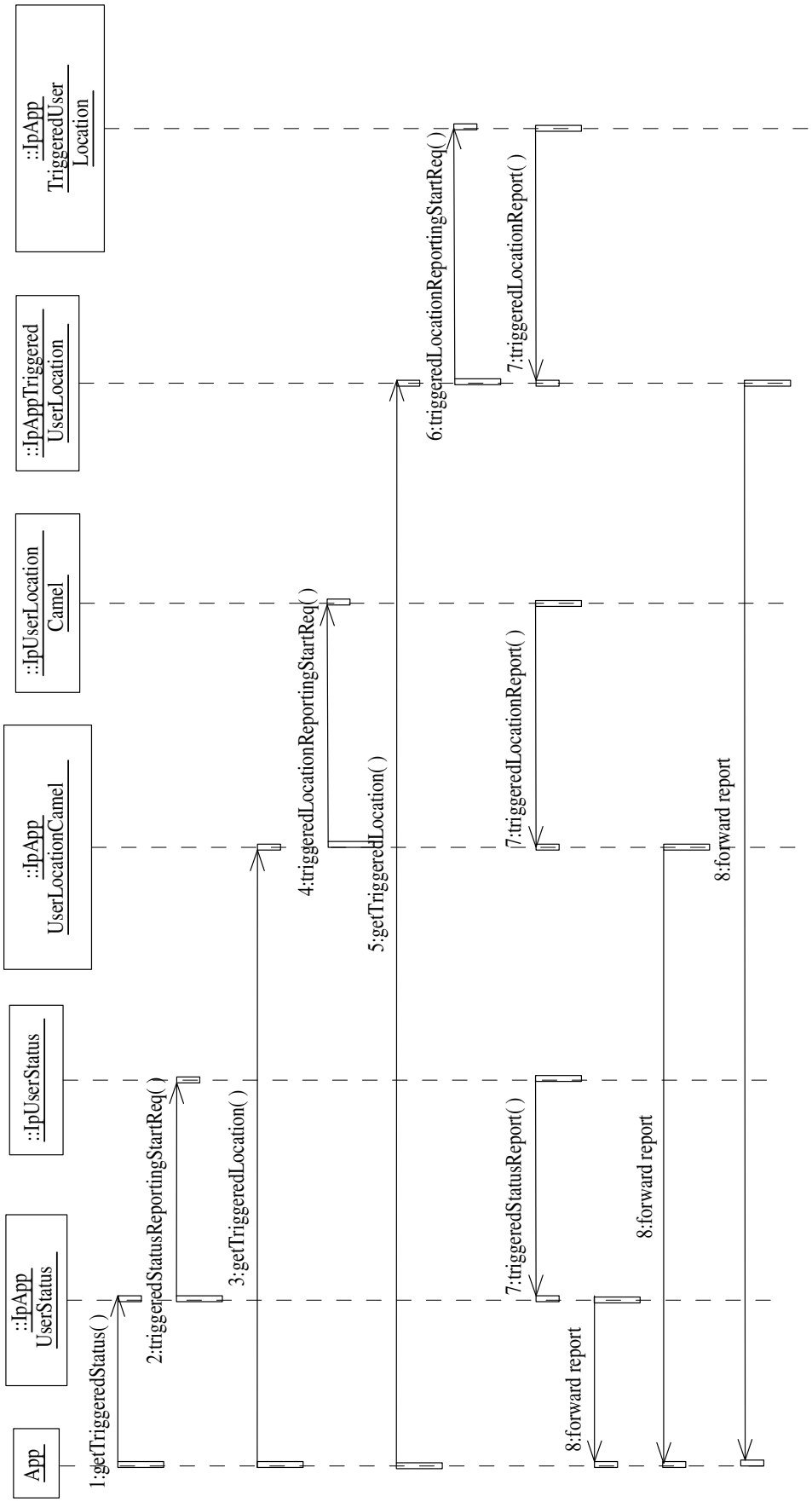

Figure 6.4: Sequence Diagram for Mobility

5. The triggered reports are returned to App.
6. This message is used to start a triggered location based on network location information, reporting for the destination terminal. The reports are returned via the *triggeredLocationReport()* method.
7. The terminal capability, user status, geographical location and network location requests are passed simultaneously and the corresponding reports are also received simultaneously.

- **Conference Call Control SCF**

1. The App creates a new object to receive the callbacks from the conference call control manager.
2. The application reserves resources for some time in the future. With this same method the application registers interest in the creation of the conference when the first party joins the conference at the specified start time. The reservation also includes the conference policy.
3. The conference is created.
4. The message is forwarded to the application.
5. The application creates an object to receive the call back messages from the conference call.
6. The application also requests to be notified when parties leave the conference.
7. The application is notified of the first party that joined the conference
8. When the party is allowed to join the conference, the party is added.
9. Alternatively, the party could have been rejected with a *releaseCallLeg()*.
10. A new party joins the conference and the application is notified.
11. The message is forwarded to the application.
12. This party also is allowed into the conference by attaching the leg.
13. A party leaves the conference.
14. The message is forwarded to the application.
15. The application decides to release the entire conference.

6.4 Chapter Summary

In this chapter, we looked at the application of the service federation and service composition. An audio conference service was used to illustrate service federation. The combination

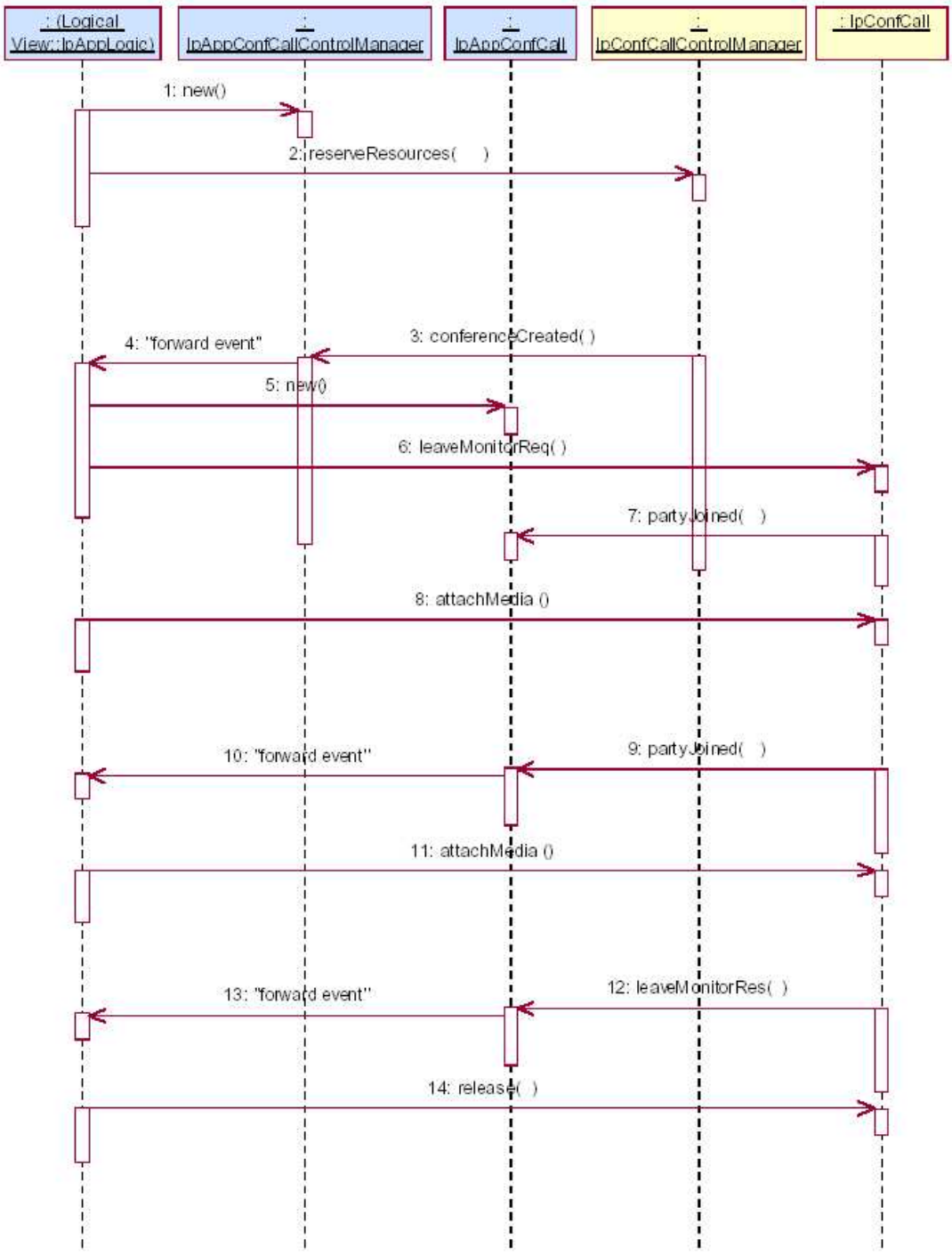

Figure 6.5: Sequence Diagram for Conference Call SCF

of the audio conference service with a shared whiteboard service was used to illustrate service composition. Service federation was used together with service composition. The terminal capability, conferencing, mobility SCFs were used for the conferencing and retrieval of terminal network information, which is needed for a reliable service federation.

Chapter 7

Conclusion

7.1 Discussion

In this report, we set out to discuss the high level design of a *teleworking service*, which consists of integrated simpler services like the e-mail, shared whiteboard, telephony, and conferencing. We started by giving a background knowledge on teleworking service in general, NGN, and the OSA / Parlay architecture. We then defined the objectives of this research and the intended approach to realising the objectives. The need for a new teleworking service, considering that there are existing forms of teleworking service, was discussed. This teleworking service is designed with an integration of different simpler services, QoS specification by application and high security.

Two essential aspects of the teleworking service were identified. First we identified the need for teleworkers in the same and different ASP domains to interact with each other. We discovered that the ASPs in different domains will have to interwork based on an agreement to achieve this service interworking. We referred to this as *service federation* i.e., the interworking of ASPs services based on an agreement. We looked into how service federation can be implemented within the OSA / Parlay environment via the use of the OSA / Parlay framework. This led to the term, framework federation, interworking of OSA / Parlay frameworks.

Second, we identified the need to combine / reuse the listed simpler services to form a new service instance, within the same service session. This we referred to as *service composition*. We looked at achieving service composition via the use of trading services. The federation between trading services, i.e., traders in different ASP domains, was described together with database federation.

7.2 Conclusions

This report looked into the concepts used to achieve service federation and service composition. Framework federation was used for the realisation of service federation. We defined framework federation as the interworking of frameworks to retrieve terminal information and locate SCFs for the purpose of service interworking. Considering that communication across networks entails using of SCFs in another ASP's domain, we looked for a way in which the application can gain access to the SCFs. These were all achieved via the federation of OSA / Parlay frameworks.

Security was achieved through framework authentication. The frameworks have to be authenticated by each other before any type of communication can occur between them. The application also has to be authenticated and authorised by the framework before accessing and using the SCFs. The OSA / Parlay APIs are also secure.

The retrieval of terminal location was found to be essential in determining whether a reliable QoS communication can occur between two terminals. QoS was realised by the application being able to compare terminal capabilities to decide whether communication can occur at a high enough QoS. As part of service usage, the teleworker is given the opportunity to choose some QoS levels, which are easy to understand by the layman, for the basic services or subscription type. The ASP makes various end-users' applicative services perform with the QoS stipulated in the service contracts. QoS signalling and control is necessary to ensure that the various levels of QoS as requested by the teleworker are met. [23] gives a formalisation of this within the framework.

Due to the limitations of the present OSA / Parlay specifications in terms of interactions between frameworks, we introduced new framework interfaces to permit the exchange of requests between frameworks. With the introduction of the new framework interfaces, we found that some already existing methods of the trust and security interfaces can be reused for framework to framework authentication and authorisation. This saves the time of thinking up new methods and parameters.

Trader federation was concluded to be the way to achieve service composition. A federation between trader services in different ASP domains allowed teleworkers to combine two or more services, which are offered by different ASPs and construct a new service. The traders return the list of services together with the object references. The application gets the service reference and accesses the required service. The retrieval of the service reference introduced the need for service provider federation. Therefore, we say that service composition requires framework federation for full implementation. We looked at the addition of database federation to check user profiles that reside in different ASP domains for

service usage policies.

We illustrated service federation and service composition using the audio conference service and the shared whiteboard service. We conclude that:

- there is the need for the introduction of new framework interfaces to achieve framework federation and hence, service federation;
- framework federation allows for secure service federation with high QoS;
- trader federation can be applied to achieve service composition;
- service composition requires federation of frameworks to function properly;
- trader federation can be extended to include database federation for checking of user profiles across ASP domains;
- The teleworker uses the teleworking service without knowledge of how operations are passed from one computational object to the other. The service logic is transparent to the teleworker.

Already implemented subscription service [24] and billing service [25] on the SATINA-NGN platform can be integrated with this teleworking service. With this integration, we can have a teleworking service which consists of subscription, billing and service usage aspects.

The design of this teleworking service helps to make the use of the teleworking service easier for the teleworker. The teleworker needs not worry about configuring the system in the case of a service failure. The ASP or telco takes care of the repair of the teleworking service. The teleworking service is delivered at a specified QoS and allows for multimedia and multiparty operations. With these improvements to the teleworking service, more people will move to being teleworkers and thus enjoy the benefits of the teleworking service.

The proposed framework federation and hence, new framework interfaces is a potential addition to the present OSA / Parlay specifications.

7.3 Recommendations for Further Work

Several extensions can be made to this teleworking service. Some of the identified extensions are:

1. **Anonymous users**: This teleworking service is designed to be used only by teleworkers who have subscribed to the teleworking service through an ASP. An extension is that non-subscribers to the teleworking service can be permitted to use the teleworking service and billed on a per-use basis. This brings in more revenue for the ASP. The anonymous teleworker can log into the teleworking service and be billed accordingly. There is no need for an offline contract to exist between the ASP and the anonymous teleworker. Also, the trading service can be used to find non-subscribed users who are in different domains. The anonymous teleworker has a variety of ASPs and services to choose from instead of just a number of affordable ASPs.

2. **Service composition without service federation**: In this teleworking service, we showed that there must be a framework federation before traders in each of these domains can federate. This implies that a federation must exist between both domains before a trader federation can exist. A teleworking service can be designed to allow traders to federate without a federation existing between their frameworks. This approach creates the ability to locate more service instances and within a faster period of time since federation contracts do not have to be checked between frameworks.

3. **Billing**: Billing is an important aspect of teleworking service, which was not looked into in this research. A vital aspect of billing for the teleworking service is the distinction between the use of the teleworking service for personal and official use. The teleworker should be allowed to use the same infrastructure for personal and official use. The billing system should be designed in such a way that the teleworker is adequately billed for personal use and not the subscriber.

4. **Service Locator**: A service locator can be used to locate services for service composition. A service locator provides a mapping between a service interface identifier and a service interface address [18]. A service locator also provides a relocation service i.e. the new location of a particular service and new properties can be got from the locator without having to search for the service again. The locator gives the location of the service and saves time, i.e., less operations are passed from the application to the framework.

References

[1] Teleworking. http://ohr.gsfc.nasa.gov/family/telecommute/definitionns.htm. Last accessed 3 September 2004.

[2] TECODIS. *Deliverable D13: TECODIS' Teleworking in Co-operative Development of Industrial Software*. TECODIS, 1999.

[3] "What is Teleworking? Telecommuting". http://www.eto.org.uk/faq/defn_tw.htm. Last accessed 2 October 2003.

[4] T. Carbasho. "Companies, Employees realise the Benefits of Teleworking". *Pittsburg Business Times - In Depth: Technology*, February 23 2004. Available on www.bizjournals.com/pittsburg/stories/2004/02/23/focus4.html last accessed on 22nd of November, 2004.

[5] "What is Teleworking?". http://www.teleworker.nildram.co.uk/WhatisTeleworking.htm. Last accessed 3 September 2004.

[6] Next Generation Networks (NGN). Webproforum Tutorial by Taqua Systems, January 2001.

[7] "Moving towards the Next Generation Networks (NGN), Technical, economic and regulatory study". study summary, Arcome office for l'Autorite de regulation des telecommunications, October 2002.

[8] Ericsson. "Parlay/OSA". White paper, Ericsson mobility world.

[9] H.E. Hanrahan. "Advanced Service Architectures". ELEN 5027 lecture notes, June 2004. School of Electrical and Information Engineering, University of the Witwatersrand.

[10] API. http://optusbusiness.com.au/00/04/01/000401.asp. Last accessed 5 August 2004.

[11] J.L. Bakker and F. Panken. "Rapid development and deployment of converged services using APIs". In *Bell Labs Tachnical Journal*, volume 5 of *2*, pages 12–29, May-September 2000.

[12] D.B. Nolte C.L. Bear, W.A. Montgomery and M.C. Silva. ”Open programmable networks”. In *Bell Labs Technical Journal*, volume 5 of *3*, pages 30–41, July-September 2000.

[13] Client application. http://www.openspirit.com/OpenSpirit_Tech_Docs/v2.4.0/docs/Development/Glossary.html. Last accessed 3 September 2004.

[14] TINA-C Consortium. TINA-C Glossary of Terms. http://intranet.ee.wits.ac.za/comms/resource/TINA, January 1997. Last accessed on 20th October 2004.

[15] S. Efremidis, et al. ”TINA-oriented service engineering support to service composition and federation”. In *5th International Conference on Broadband Service and Networks (IS&N) proceedings*, Antwerp, Belgium, May 1998.

[16] S. Popov. ”Framework federation, Integrated Parlay gateways”. Technical report, Parlay Open meeting, Bangkok, 2003.

[17] A. Beitz and M. Bearman. ”Service location in an ODE”. In *2nd International Workshop on Services in Distributed and Network Environments (SDNE)*, Whistler, British Columbia, June 1995. IEEE.

[18] A. Beitz and M. Bearman. ”An ODP trading service for DCE”. In *1st International Workshop on Services in Distributed and Network Environments (SDNE)*, Prague, Czech republic, June 1994. IEEE.

[19] M.R. James, S. Mohapi, and H.E. Hanrahan. ”Using the CORBA Trading Service to Federate Distributed Databases on a Telecommunications Service Platform”. In *Southern African Telecommunication Networks and Applications Conference*, South Africa, September 2001. SATNAC.

[20] D. Mwansa and H.E. Hanrahan. ”Conceptual QoS framework for managing end-user services in the Next Generation Network”. In *Southern African Telecommunication Networks and Applications Conference*, South Africa, September 2003. SATNAC.

[21] K. Raymond. ”Reference Model of Open Distributed Processing RM-ODP: Introduction”. In *International Conference on Open Distributed Processing*, Brisbane, Australia, February 1995.

[22] *OSA/Parlay specification v4.0; Framework*. ETSI ES 202 915-3. ETSI, v1.2.1 edition, August 2003.

[23] D. Mwansa and H.E. Hanrahan. ”Formalising the Next Generation Network enduder QoS signalling and control framework”. In *Southern African Telecommunication Networks and Applications Conference*, South Africa, September 2004. SATNAC.

[24] J-C. Lee. ”Implementation of TINA Service Subscription Management using ODBMS”. Master’s thesis, School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, 2001.

[25] C-Y. Chen. ”Telecommunications service accounting based on TINA”. Master’s thesis, School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, 2002.

Appendix

Appendix A

IDL for Service Federation and Service Composition

A.1 IDL Definitions for Required Information

A.1.1 Authinfo.idl

// contains all necessary authentication information

```
interface Authinfo {

 typedef string TpAuthDomain;
 struct TpAuthType
 {
  string authType;
 };
 struct TpVersion
 {
  string frameworkVersion;
 };
 struct TpInterfaceNameList
 {
  sequence<string>interfaceNameList;
 };
 typedef string TpInterfaceName;

 exception TpCommonExceptions
```

```
 {
  string errorCode;
  string errorName;
 };
 };
```

A.1.2 RequestInfo.idl

```
//required information about the request being passed

interface RequestInfo {
 struct TpAssignmentID
 {
  string assignmentID;
 };
 struct TpuserInfo
 {
  string originatingAddress;
  string userId;
  string destinationAddress;
 };

 exception TpCommonExceptions
 {
  string errorCode;
  string errorName;
 };
 };
```

A.1.3 serviceInfo.idl

```
//represents the IDL for the information that supports the service
//i.e the required information about a service

 interface serviceinfo {
 struct TpServiceTypeName
 {
```

```
 string serviceTypeName;
 string SCFName;
};
typedef long TpInt32;
typedef Object LifeCycleManagerRef;
struct TpServiceProperties
{
 string name;
 string mode;
 string type;
};
struct TpServiceTypeDescription
{
 string name;
 boolean isavailable;
 sequence<string> serviceProperties;
};
struct TpDesiredPropertyList
{
 string name;
 sequence<string> valuelist;
};
struct TpServicePropertyList
{
 sequence<string> desiredPropertyList;
};
struct TpServiceList
{
 string serviceID;
 sequence<string> serviceProperties;
};
struct TpServiceTypeNameList
{
 string ServiceTypeNameList;
};
exception TpCommonExceptions
{
 string errorCode;
 string errorName; };  };
```

A.2 IDL for The Framework

A.2.1 IDL for Framework Request Interface

```
#include "serviceinfo.idl"
#include "requestInfo.idl" #include
"Authinfo.idl"
module Framework {
 typedef Object ObjectRef,
 serviceInterfaceIdentifier;

 interface IpFwRequest
 {
  void invitationRequest
       (in RequestInfo::TpuserInfo userId,
        in RequestInfo::TpuserInfo originatingAddress,
        in RequestInfo::TpUserInfo destinationAddress,
        in serviceinfo::SCFName)
 //This parameter is optional when passing the request to Fw2
 raises(RequestInfo::TpCommonExceptions);

 //request for a new  service instance
 serviceInfo::TpServiceList newServiceReq
 (in serviceinfo::TpServiceTypeName serviceTypeName,
  in serviceinfo::TpServiceProperties name,
  in serviceinfo::TpServicePropertyList desiredPropertyList)

  raises (serviceinfo::TpCommonExceptions);

//request for reference to the selected service
 serviceInterfaceIdentifier getServiceRef
 (in serviceInfo::TpServiceTypeName serviceTypeName,
  in ObjectRef)

 raises (serviceinfo::TpCommonExceptions);
```

```
//request to use SCFs in another provider’s domain.
 requestInfo::assignmentID useSCF
 (in RequestInfo::TpuserInfo userId,
  in RequestInfo::TpuserInfo originatingAddress,
  in requestInfo::TpUserInfo destinationAddress
  in serviceinfo::TpServiceTypeName SCFName)
  raises (requestInfo::TpCommonExceptions);

//request to know the SCFs supported by the framework.
serviceinfo::TpListTypes listInterfaces( )
raises(serviceinfo::TpCommonExceptions);
}; };
```

A.2.2 IDL for Framework Network Information Interface

```
module Framework { interface IpFwNetworkInfo
{ void getNetworkInfo
 (in serviceinfo::TpServiceTypeName SCFName,
  in requestInfo::TpUserInfo destinationAddress,
  in RequestInfo::TpuserInfo originatingAddress)

raises(requestInfo::TpCommonExceptions);
}; };
```

A.2.3 IDL for Framework Authentication

```
module Framework {
 interface IpInitial
 {void initiateAuthenticationWithVersion(
       in AuthInfo::TpAuthDomain domain,
       in AuthInfo::TpAuthType authType,
       in AuthInfo::TpVersion frameworkVersion)

  raises (AuthInfo::TpCommonExceptions);
}; };
```

Appendix B

IDL for SCFs

B.1 IDL Definitions for SCF Common Data

B.1.1 Terminal Capability Data

```
//data that support Terminal capability requests
interface Tcapdata {
 typedef long TpAssignmentID;
 typedef Object IpAppExtendedTerminalCapabilitiesRef;

 struct TpAddress
 {
  string Plan;
  string AddrString;
  string Name;
  string Presentation;
  string Screening;
  string SubAddressString;
 };

 typedef sequence<TpAddress> TpAddressSet;
 //holds the device addresses and address properties

 typedef string TpTerminalCapabilityScope,
                TpTerminalCapabilityChangeCriteria;

 struct TpTerminalCapabilities
```

```
 {
  sequence<string> capability;
  sequence<boolean> isavailable;
 };

 exception TpCommonExceptions
 {
  string erroCode;
  string errorName;
  string errorValue;
 };
 };
```

B.1.2 Mobility Data

```
//data that support mobility SCF
interface mobility {
 typedef Object IpAppTriggeredUserStatusRef;

 struct TpAssignmentID
 {
  long assignmentID;
 };
 struct TpUserStatus
 {
  string userId;
  string statusCode;
  string status;
  string terminalType;
 };
 typedef sequence<TpUserStatus> TpUserStatusSet;
 struct TpAddress
 {
  string Plan;
  string AddrString;
  string Name;
  string Presentation;
  string Screening;
```

```
  string SubAddressString;
 };
 typedef sequence<TpAddress> TpAddressSet;

 exception TpCommonExceptions
 {
  string erroCode;
  string errorName;
  string errorValue;
 };
 };
```

B.2 SCF Interfaces

B.2.1 Terminal Capability SCF

```
#include "Tcapdata.idl"
//starts monitoring of the user’s terminal capabilities
 module TerminalCapability {
 interface IpExtendedTerminalCapabilities
 {
  Tcapdata::TpAssignmentID triggeredTerminalCapabilityStartReq
  (in Tcapdata::IpAppExtendedTerminalCapabilitiesRef
               appTerminalCapabilities,
  in Tcapdata::TpAddressSet terminals,
  in Tcapdata::TpTerminalCapabilityScope capabilityScope,
  in Tcapdata::TpTerminalCapabilityChangeCriteria criteria)

  raises (Tcapdata::TpCommonExceptions);

  void triggeredTerminalCapabilityStop
  (in Tcapdata::TpAssignmentID assignmentID)

  raises (Tcapdata::TpCommonExceptions);
};
 // The method above is used to stop notification reporting
//for a specific device(s)
```

```
 interface IpAppExtendedTerminalCapabilities
 {
  void terminalCapabilityReport
  (in Tcapdata::TpAddressSet users,
  in Tcapdata::TpTerminalCapabilityChangeCriteria criteria,
  in Tcapdata::TpTerminalCapabilities capabilities);
 // the terminalCapabilityReport gives the notification of
 //a network update for specific device(s)
 };  };
```

B.2.2 Mobility SCF

```
#include "mobilitydata.idl"

module mobility{ interface IpUserStatus
{mobility::TpAssignmentID
triggeredStatusReportingStartReq(
  in mobility::IpAppTriggeredUserStatusRef appStatus,
  //reference to call back interface
  in mobility::TpAddressSet user);

 //register for notifications of changes in user's status
 //and send to callback interface, IpAppUserStatus.

}; interface IpAppUserStatus {
void triggeredStatusReport(
    in mobility::TpAssignmentID assignmentID,
    in mobility::TpUserStatus status)
    //status of the specified user

    raises(mobility::TpCommonExceptions);
 //delivery of report that indicates that a user's
 //status has changed
 };
 };
```

Appendix C

Sequence Diagrams for the Application

C.1 Service Discovery

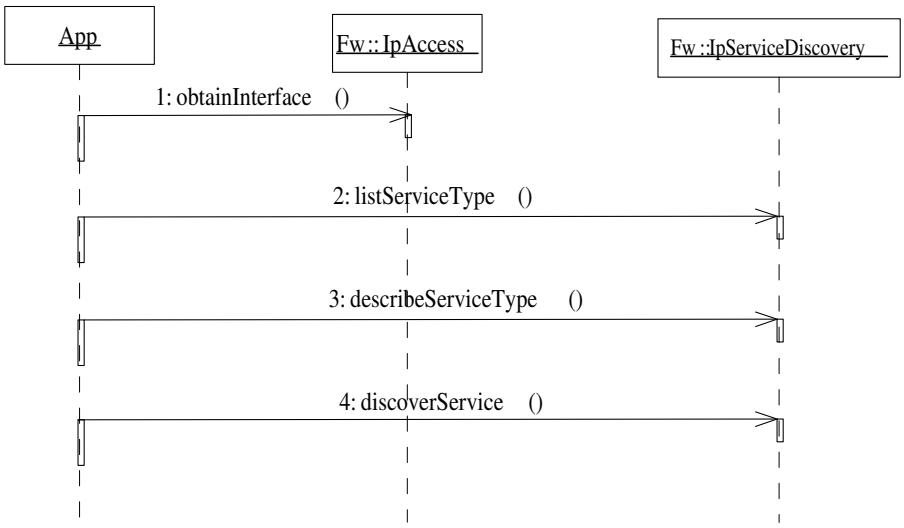

Figure C.1: Sequence Diagram for Service Discovery

1. Application invokes an *obtainInterface()* operation on framework's *IpAccess* interface. With this operation, the application requests for the reference to the framework's service discovery interface.

2. The App asks Fw for the available service types on its network. The App uses this operation to know the SCFs that are supported by the network.

3. The App requests for the properties of desired SCFs or service types. The App gives the service name (SCF name) and serviceTypeDescription (property names, property value types, etc).

4. App requests for a service that matches its needs. The framework checks for a match and sends the serviceID to App. Input parameters are the serviceTypeName and desiredPropertyList. Output parameter is the serviceList.

C.2 Authentication of the Application by the Framework

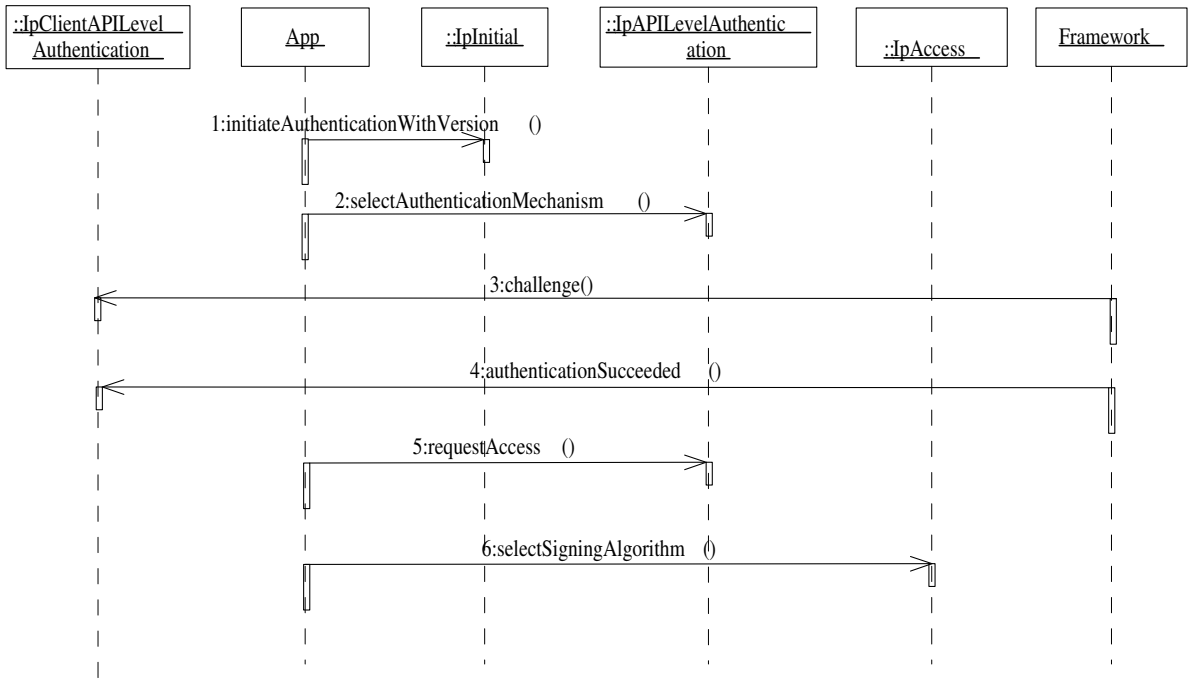

Figure C.2: Sequence Diagram for Authentication of the Application

1. The application invokes the *initiateAuthenticationWithVersion()* operation on the framework's interface to initiate the authentication process. It provides in turn a reference to its own authentication interface. The framework returns a reference to its authentication interface.

2. The application identifies the authentication algorithm it supports. The framework does not have to accept this algorithm. The mechanism to be used is prescribed by the framework.

3. The framework authenticates the application. The framework supplies a challenge and the application returns the response. The client does not have to authenticate the framework immediately after been authenticated itself.

4. The framework forwards an indication as per the success of the authentication.

5. If the authentication was successful, the application requests for the reference to the framework's access interface. the application in turn provides the framework with the reference to its own access interface.

6. The application and framework negotiate the signing algorithm to be used for any signed exchanges.